Deep Learning

If not stated otherwise graphs and code examples were taken with permission from Michael Nielsen's online book Neural Networks and Deep Learning.

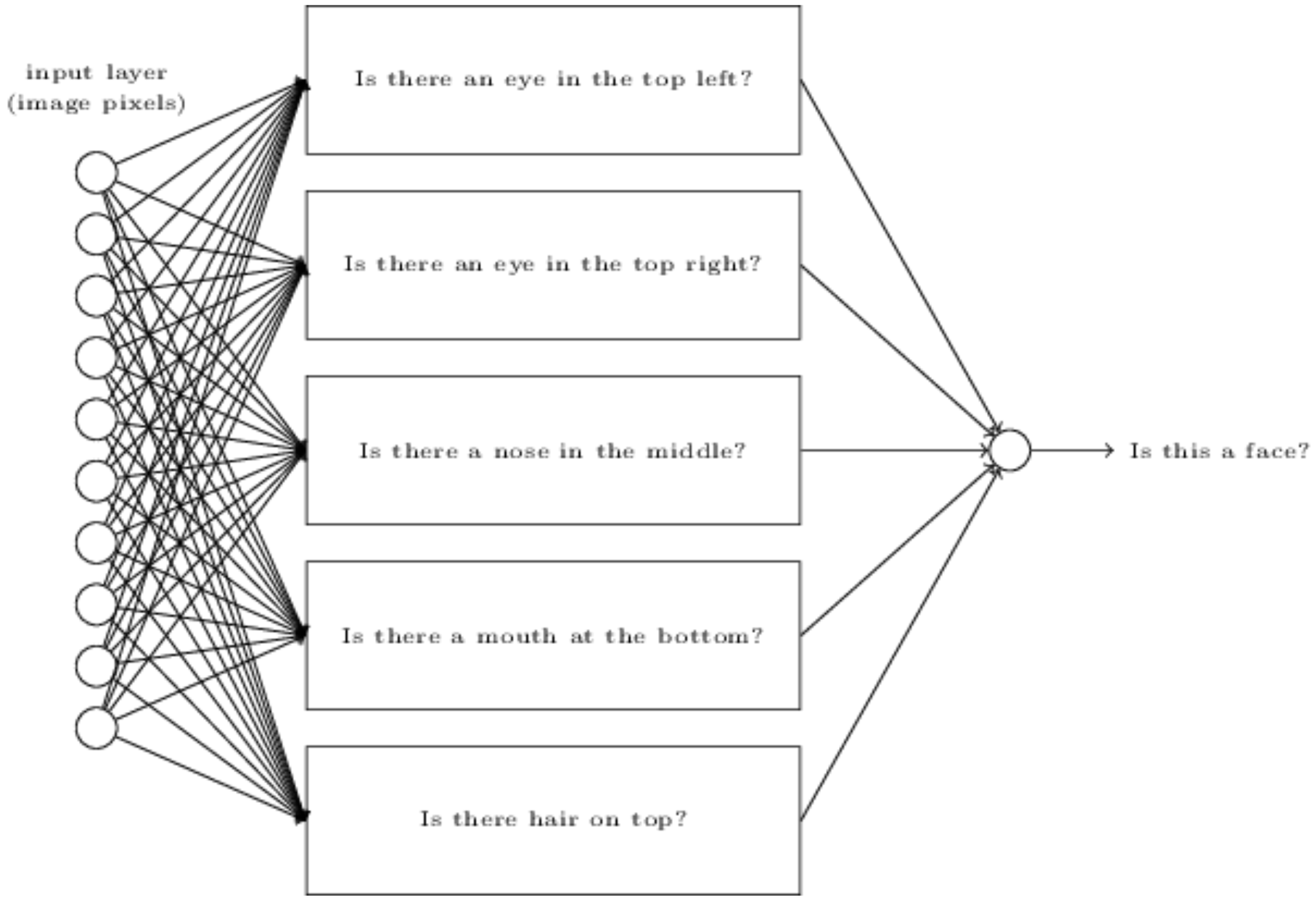
Deep learning

- is a host of statistical machine learning techniques
- is generally based on artificial neural networks
- enables the automatic learning of feature hierarchies

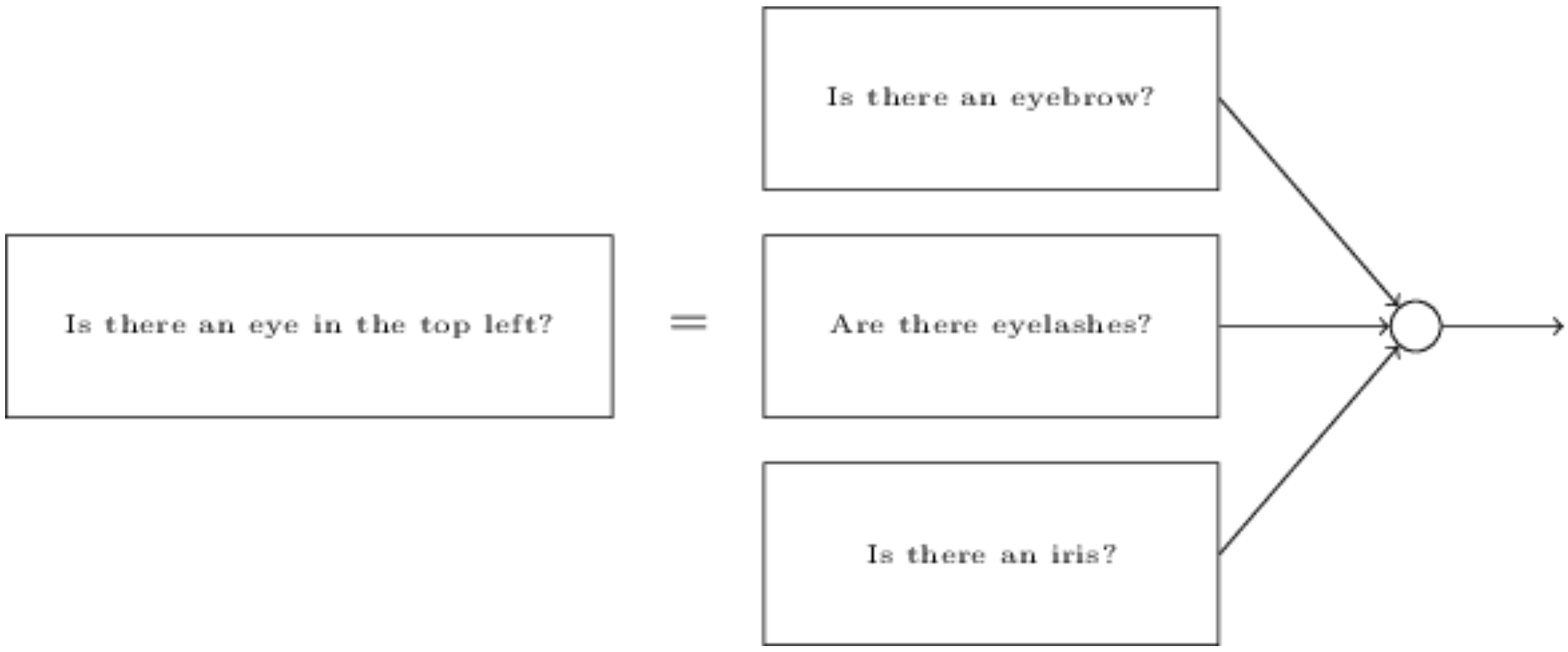
Deep learning

- is a host of statistical machine learning techniques
- is generally based on artificial neural networks
- enables the automatic learning of **feature hierarchies**

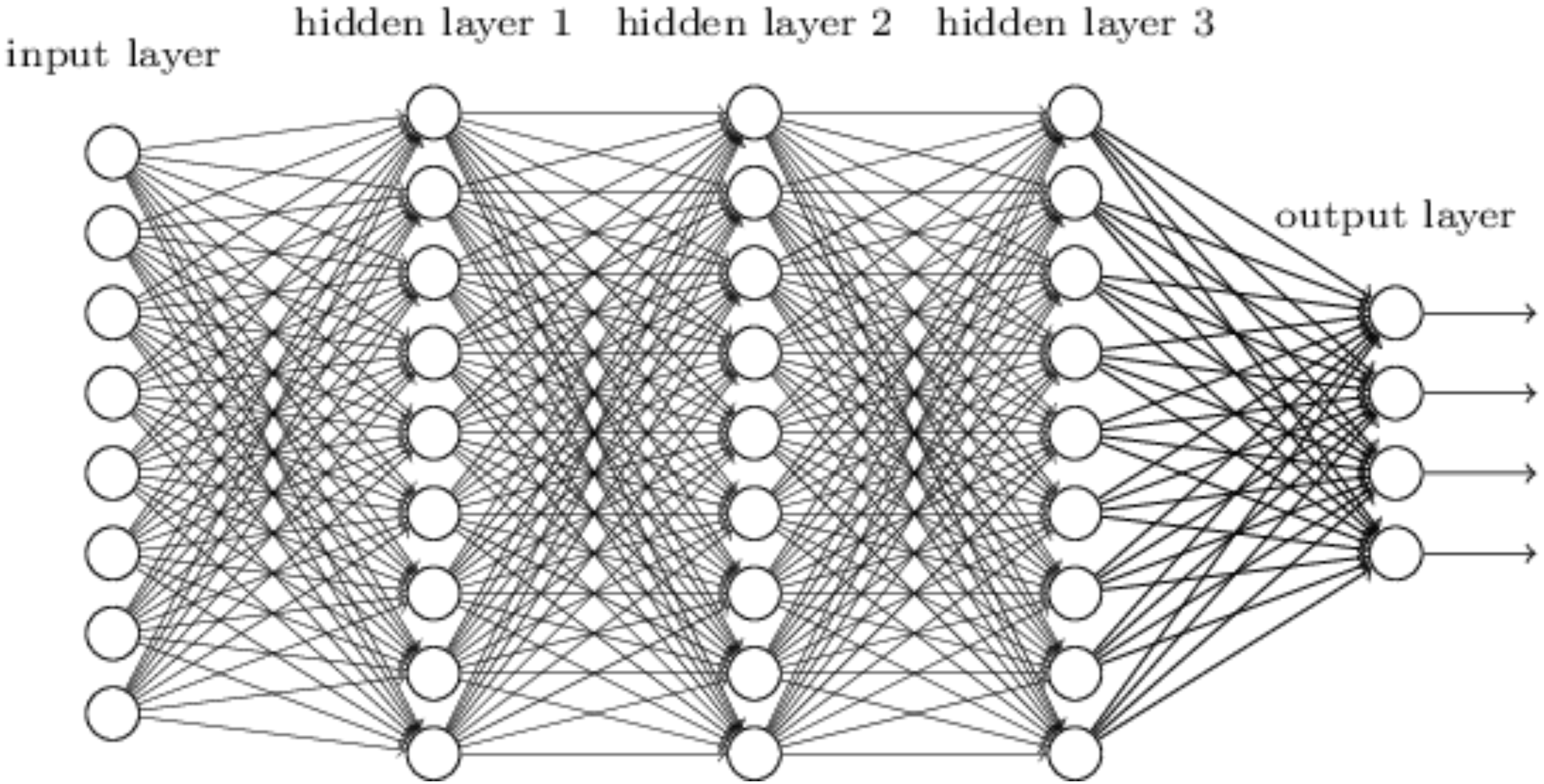
Feature hierarchy



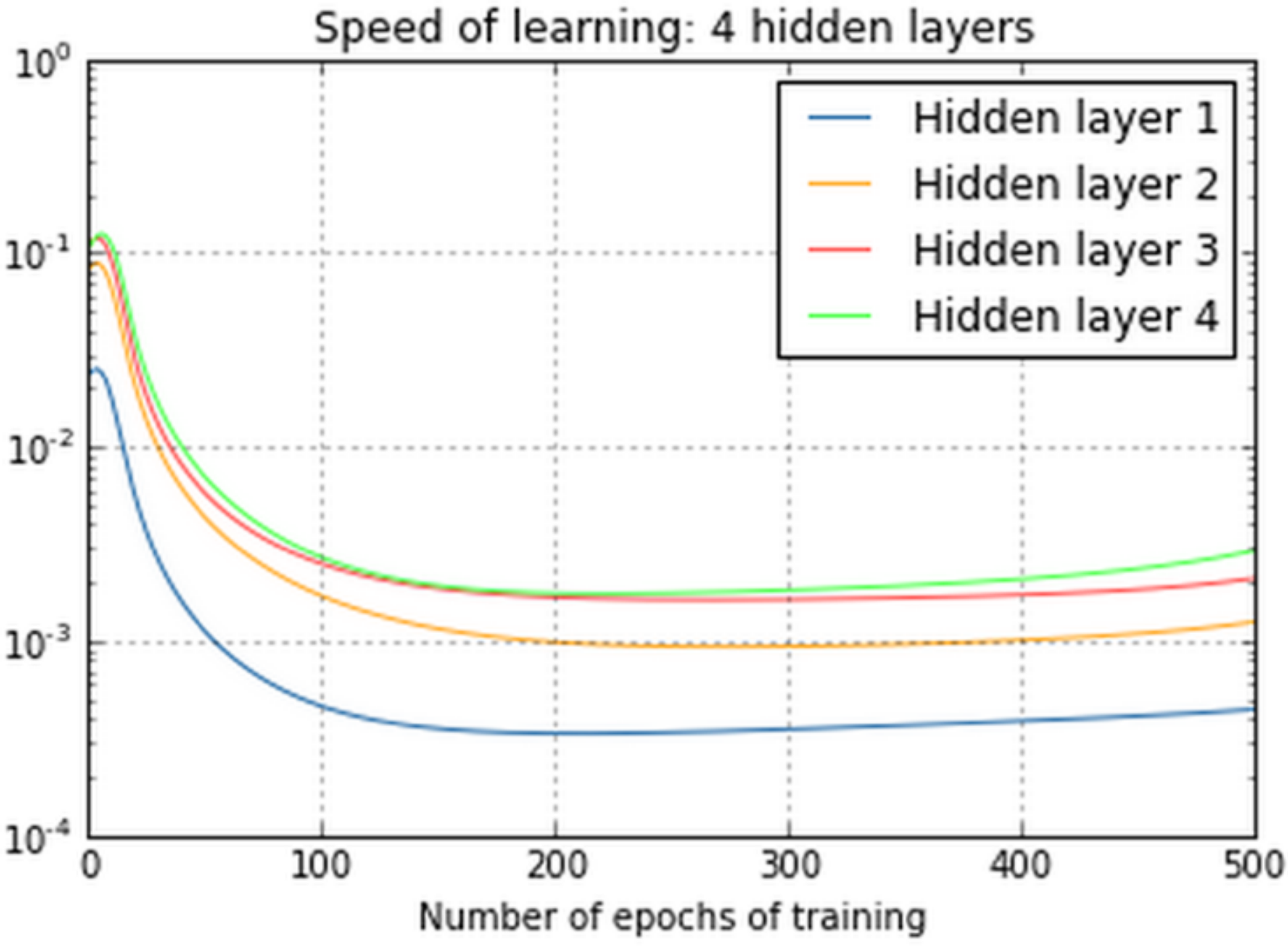
Feature hierarchy



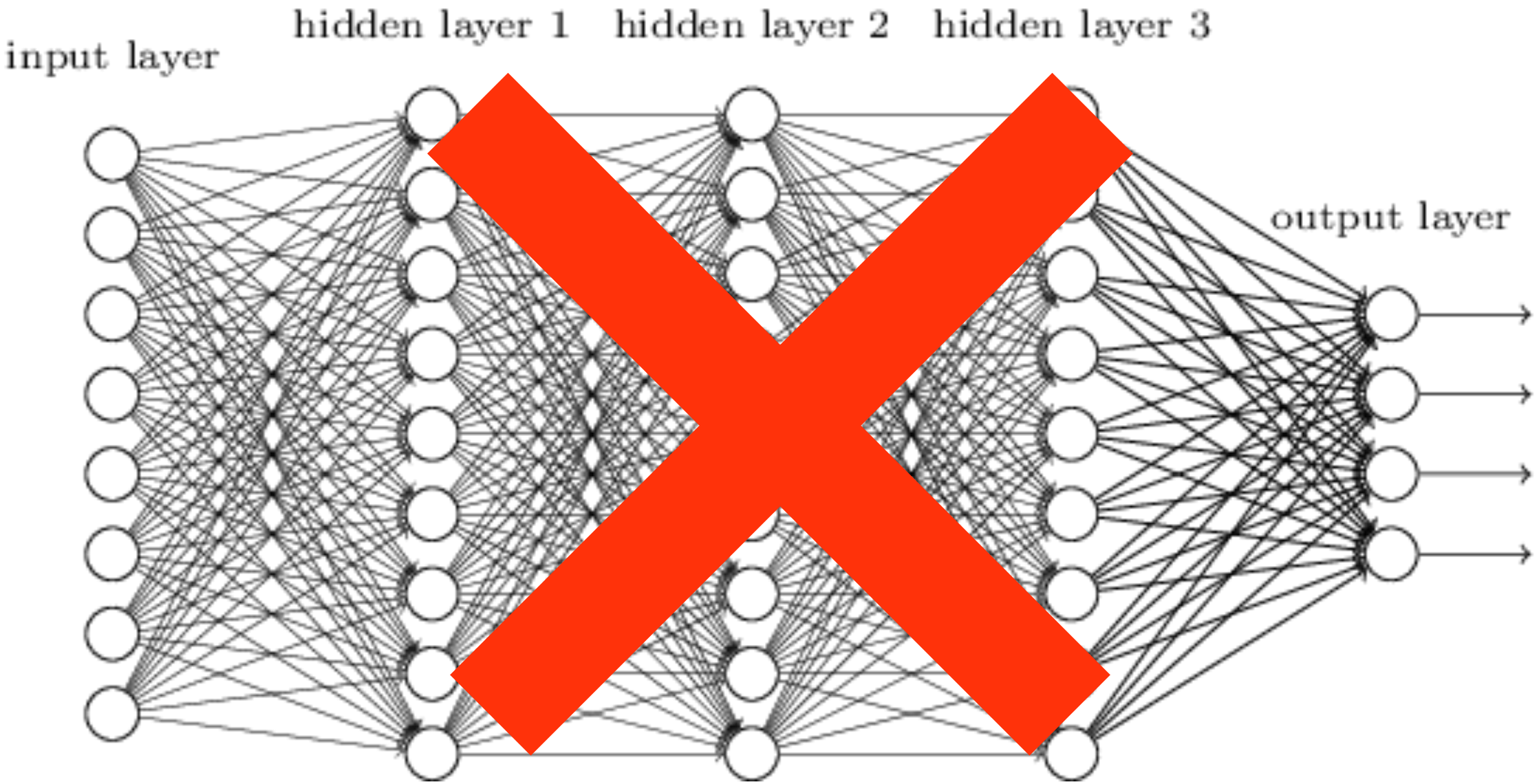
Let's train our first deep network



Vanishing Gradient Problem

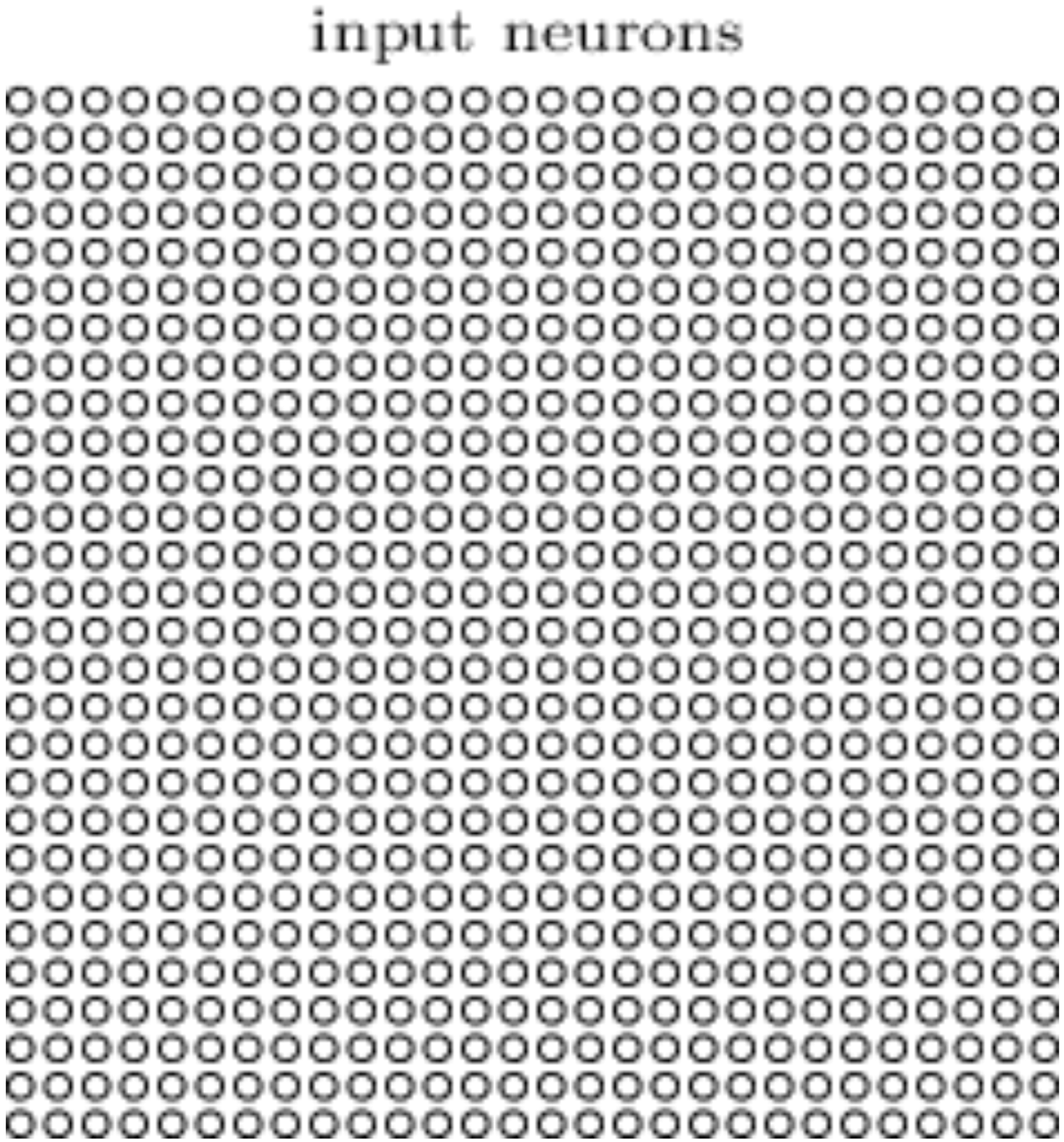


Vanishing Gradient Problem

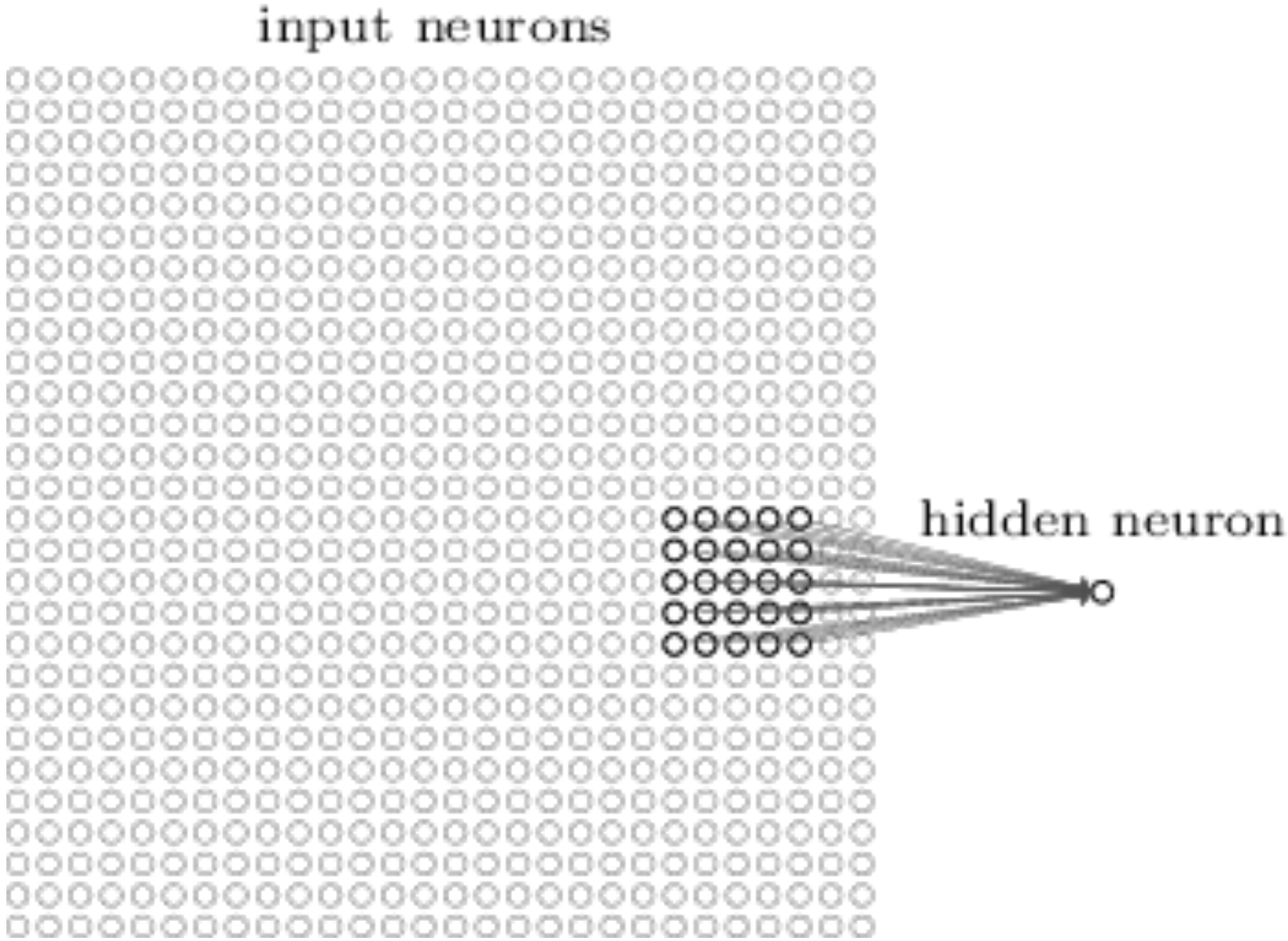


What now?

Images as Input



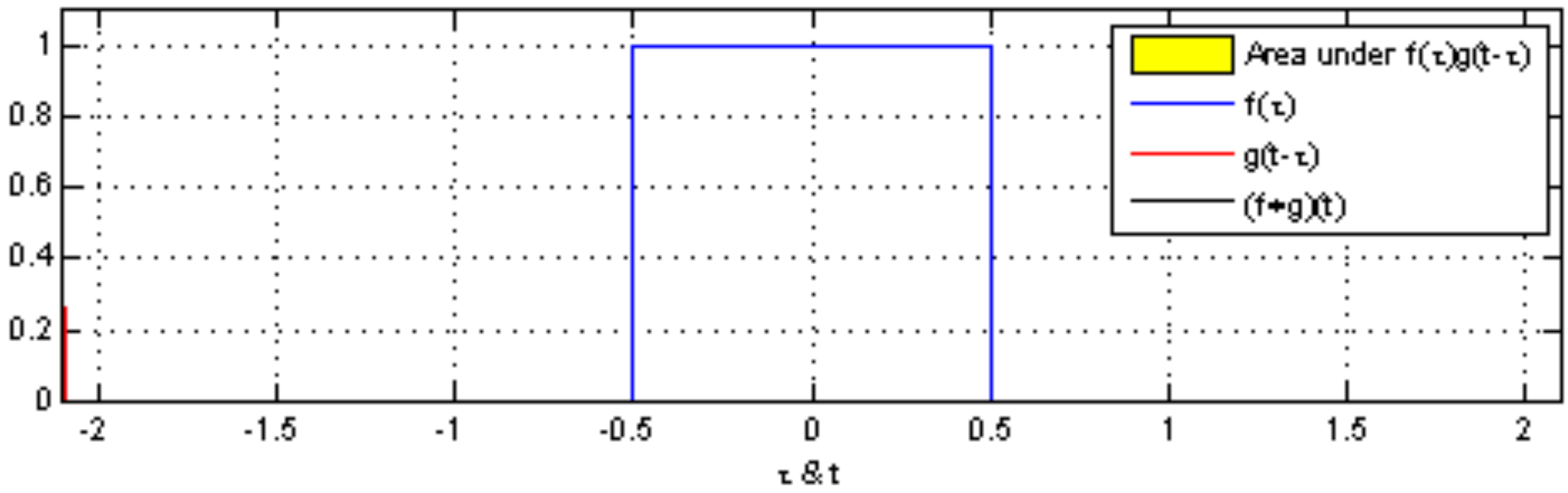
Local receptive Fields



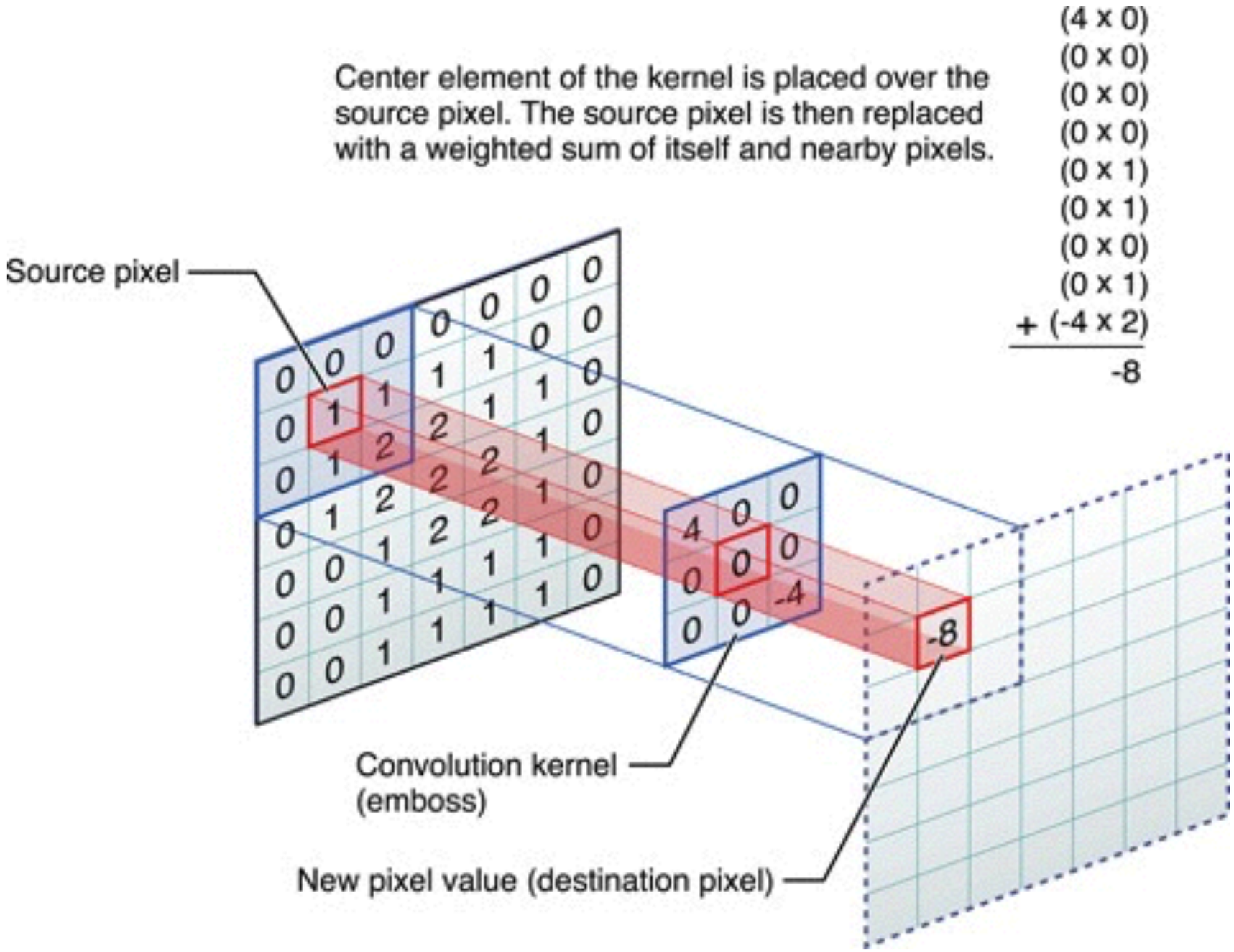
Detour Convolution

Convolution

- is a mathematical operation of two functions
- and computes the integral
- of the point wise multiplication of the two functions.

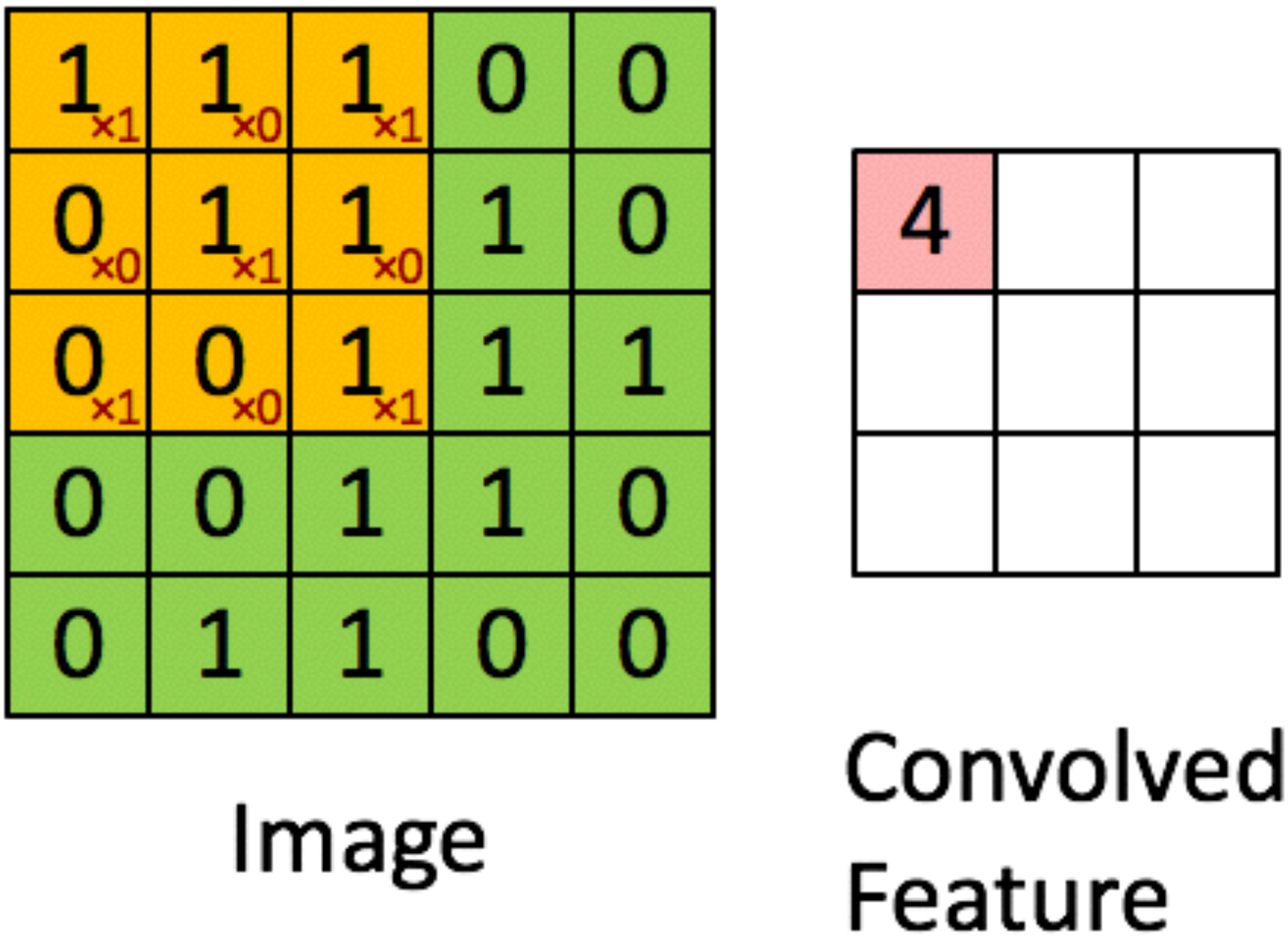


Convolution



http://www.westworld.be/convolution-kernels/

Convolution



http://stackoverflow.com/questions/21743871/how-to-perform-iterative-2d-operation-on-4d-numpy-array

Convolution

Input image

Convolution Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Feature map

http://timdettmers.com/2015/03/26/convolution-deep-learning/

Convolution

Input image

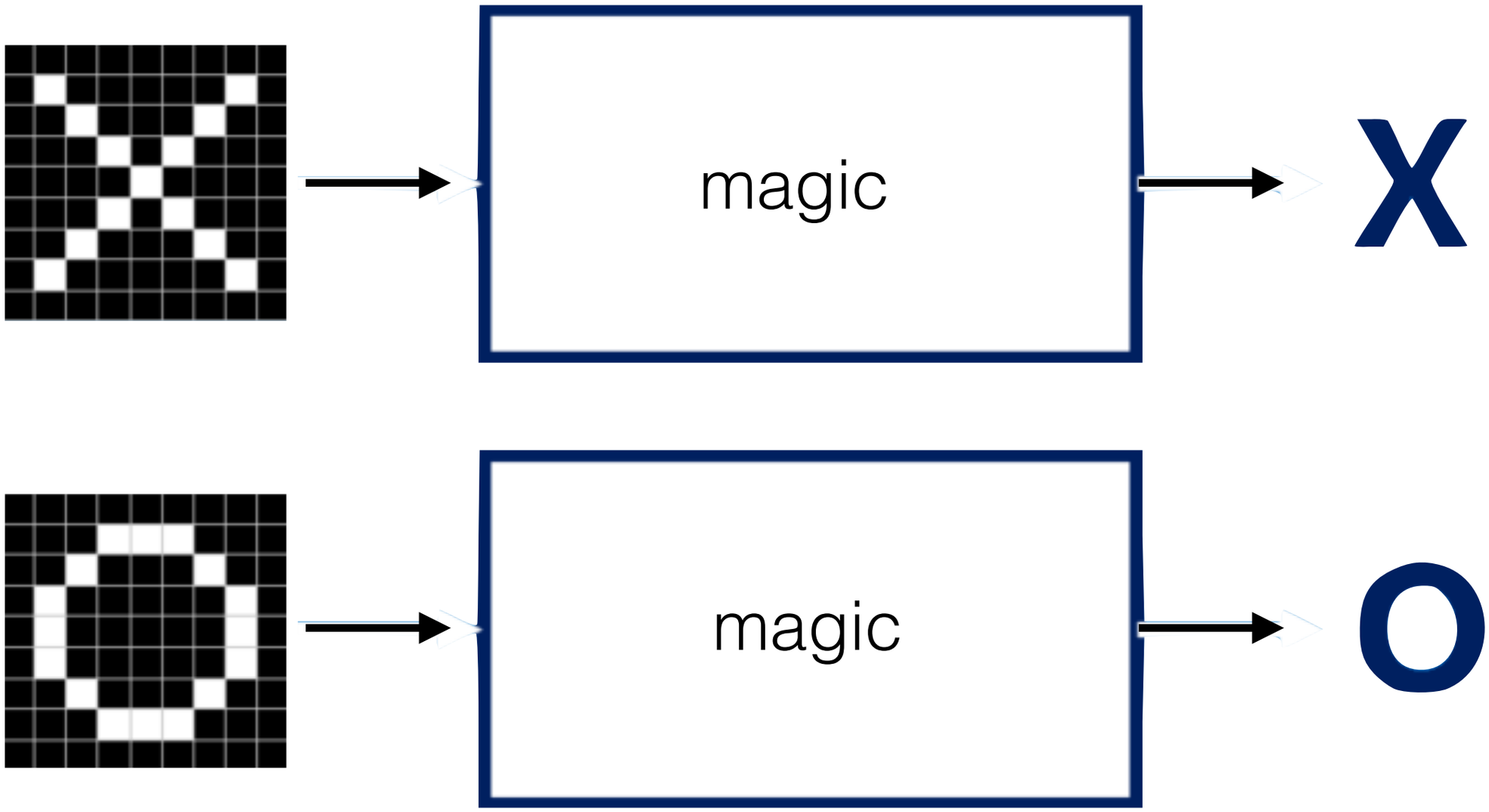
Kernel

$$\begin{pmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{pmatrix}$$

Feature map

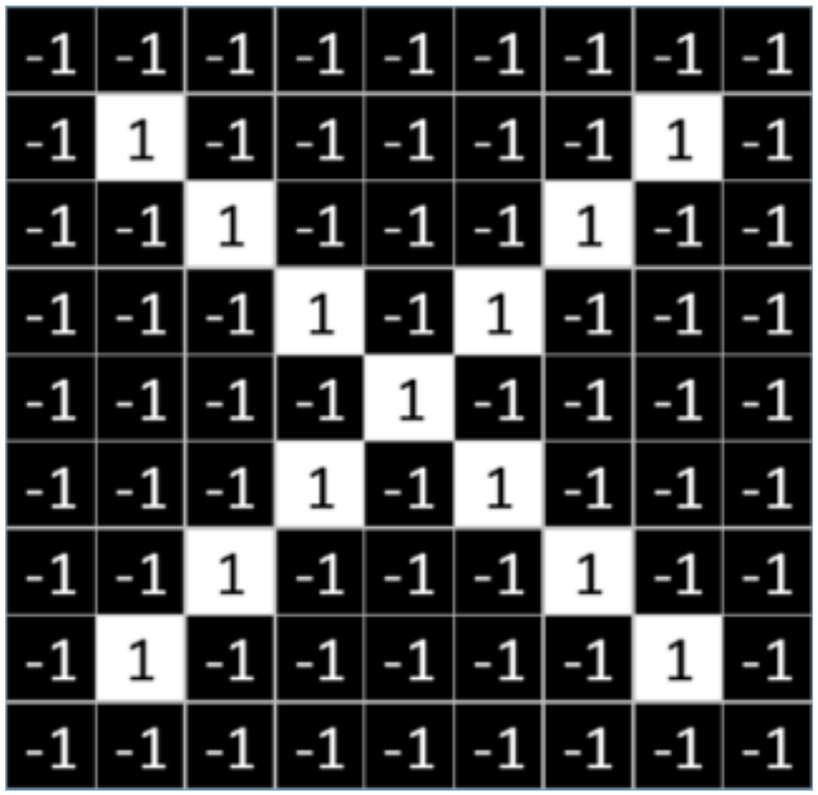
http://timdettmers.com/2015/03/26/convolution-deep-learning/

Object detection with convolutions

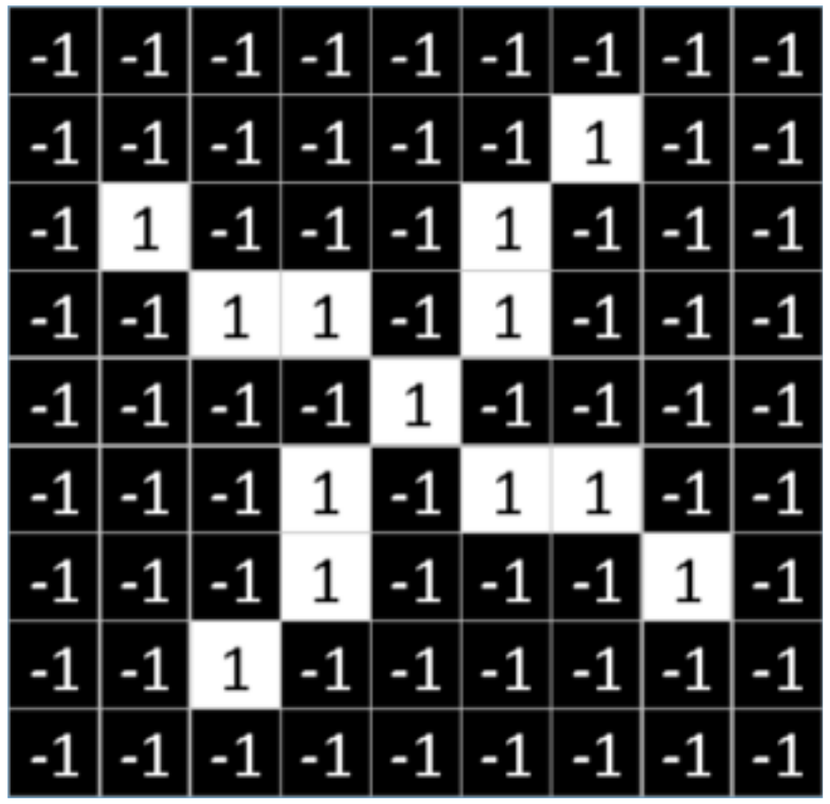


http://brohrer.github.io/how_convolutional_neural_networks_work.html

Object detection with convolutions



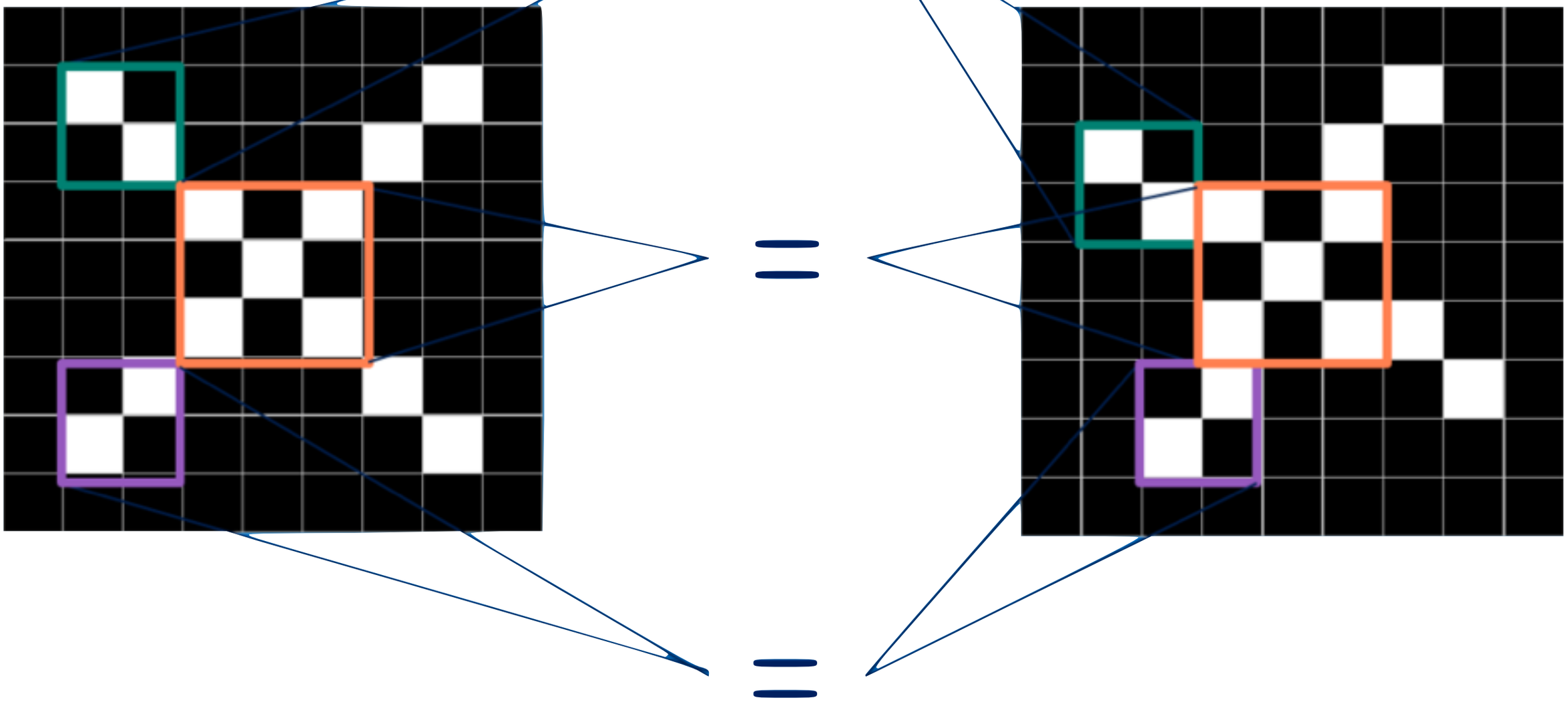
?



http://brohrer.github.io/how_convolutional_neural_networks_work.html

Object detection with convolutions

=

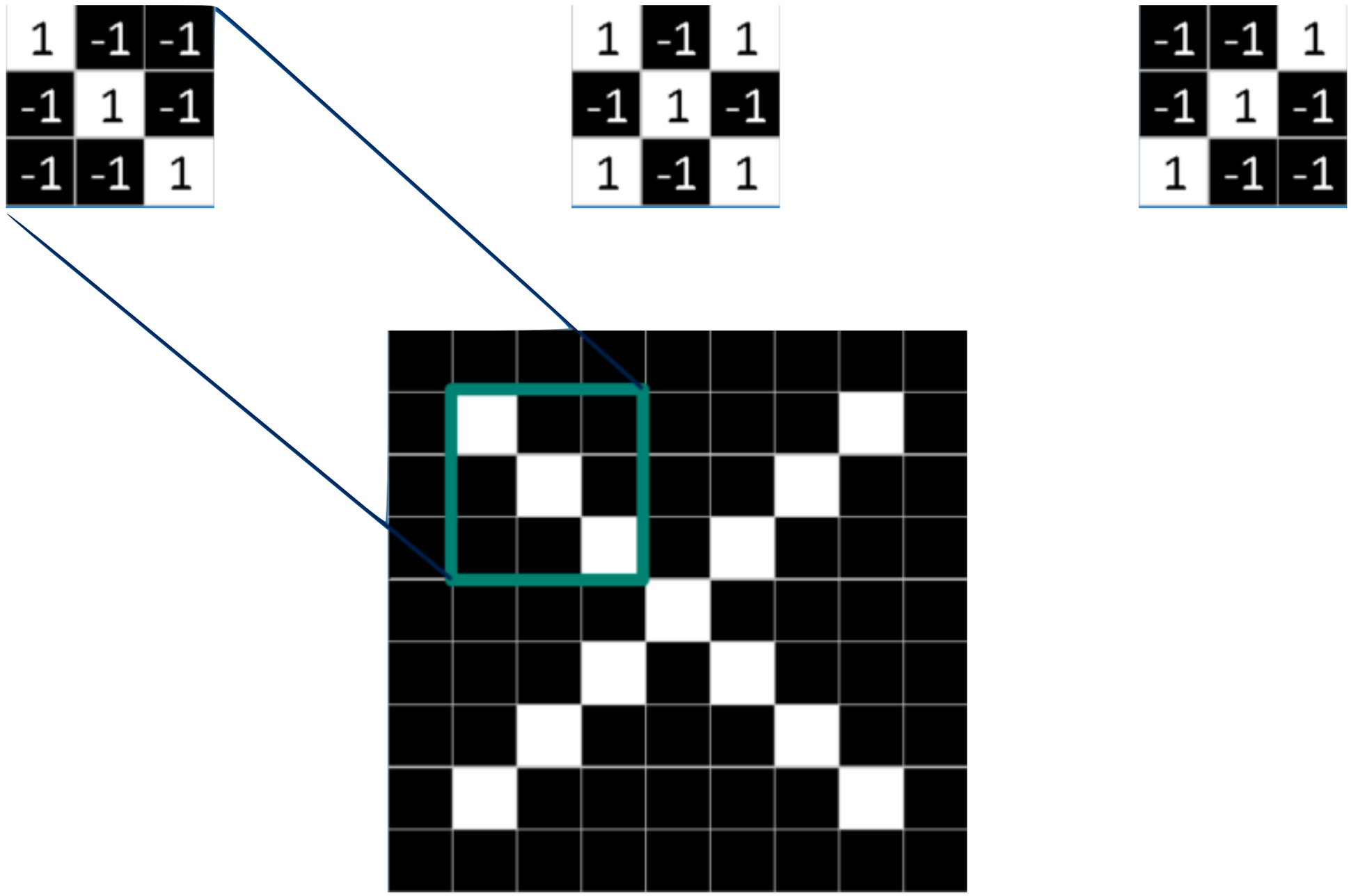


=

=

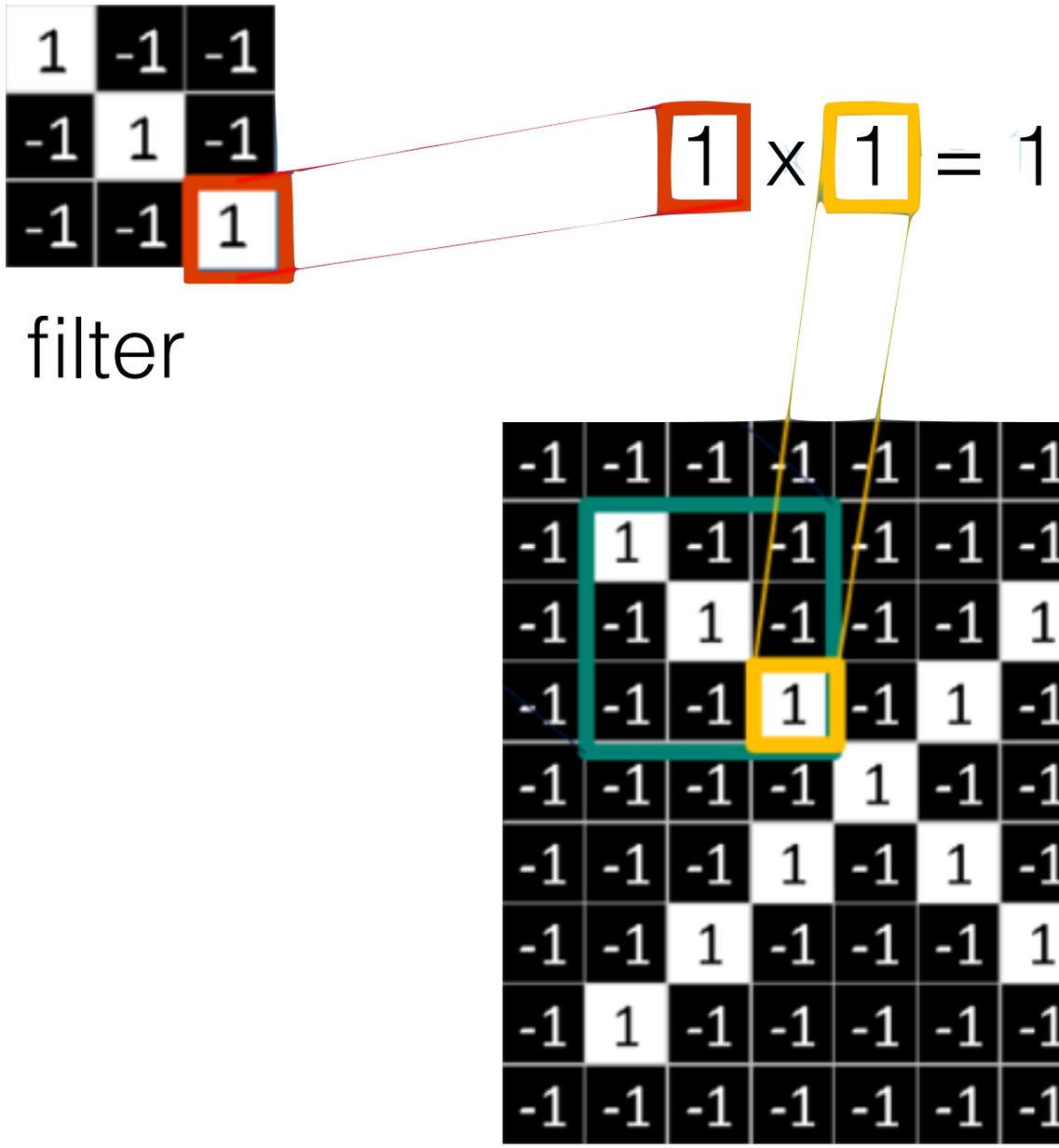
http://brohrer.github.io/how_convolutional_neural_networks_work.html

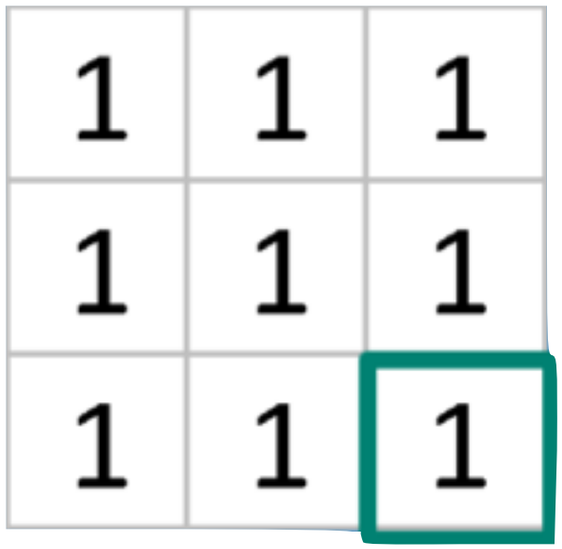
Object detection with convolutions



http://brohrer.github.io/how_convolutional_neural_networks_work.html

Object detection with convolutions

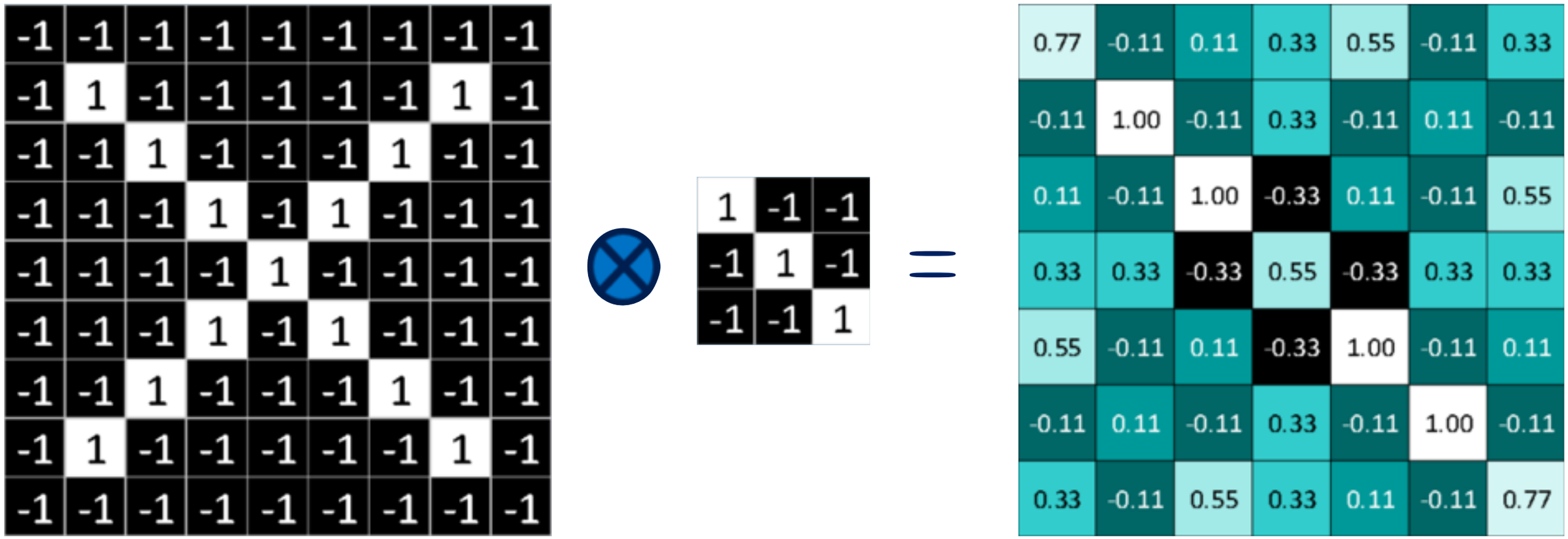




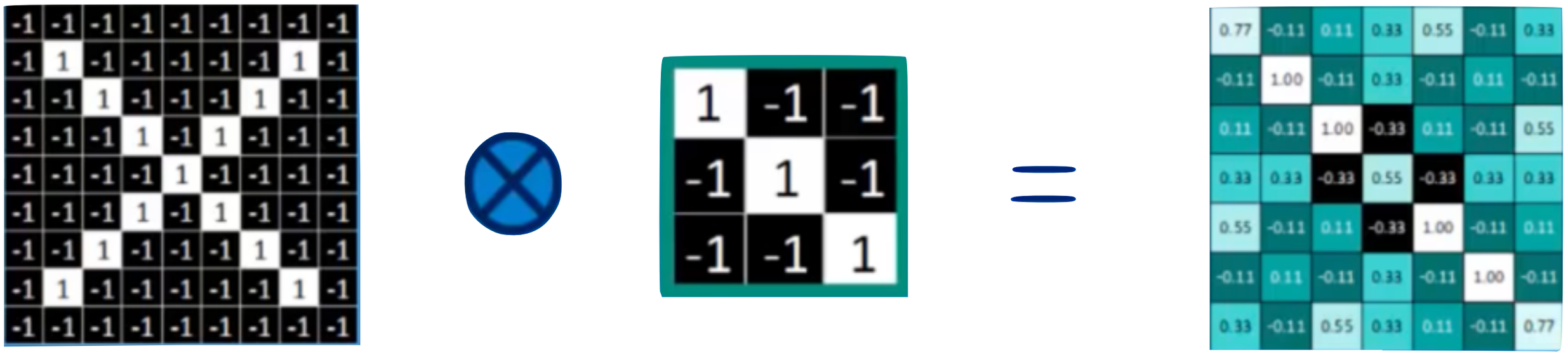
result

http://brohrer.github.io/how_convolutional_neural_networks_work.html

Object detection with convolutions

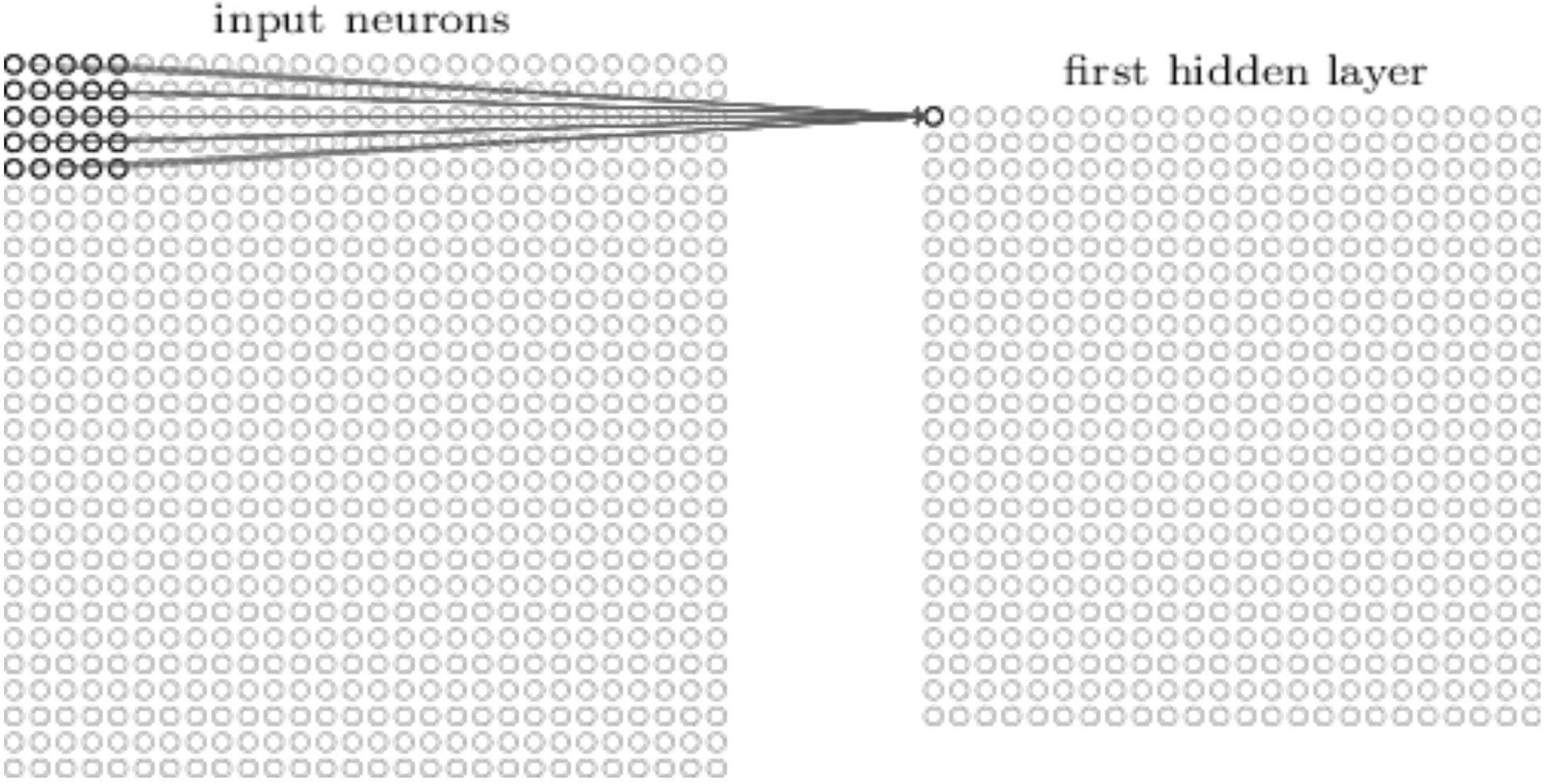


http://brohrer.github.io/how_convolutional_neural_networks_work.html

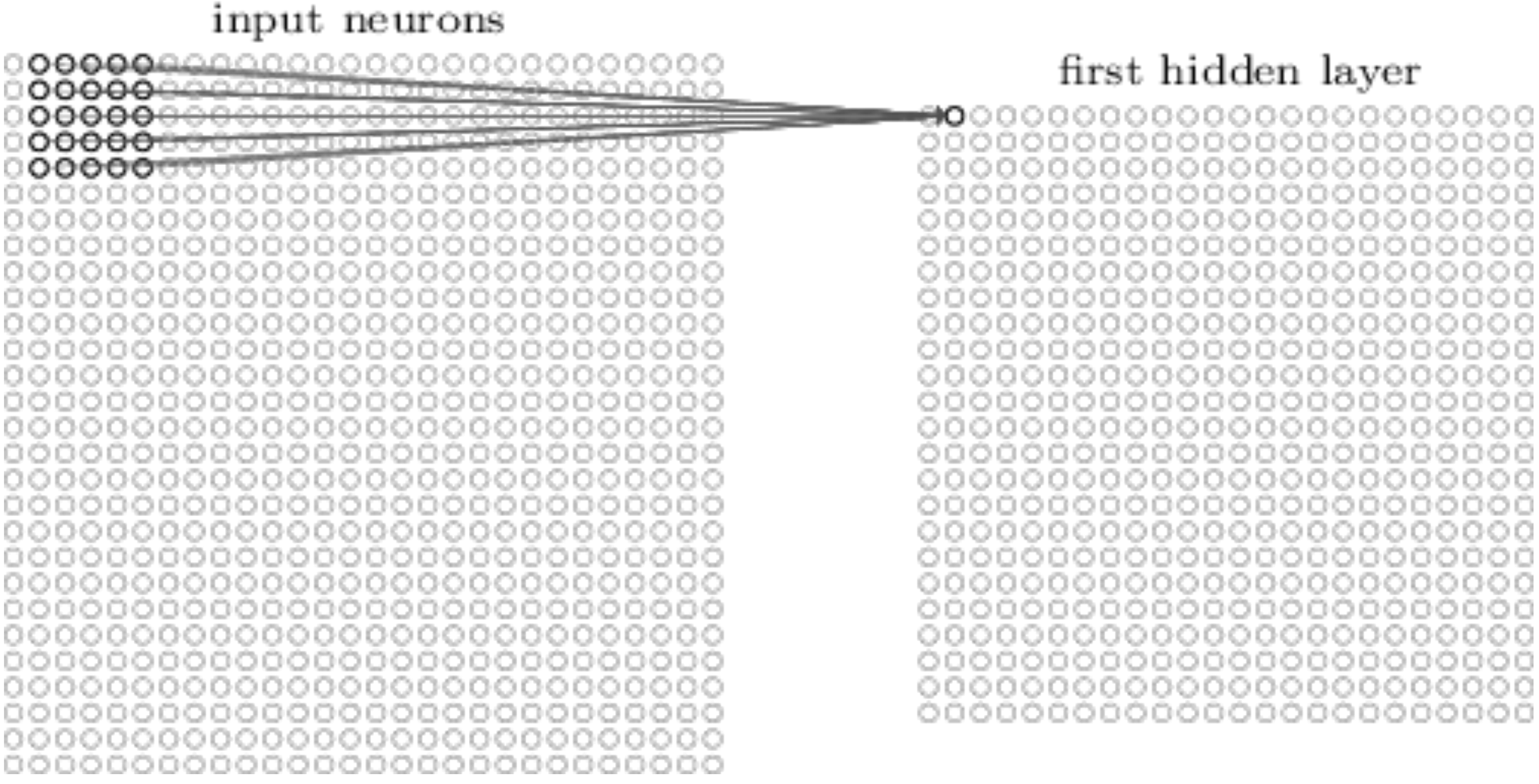


http://brohrer.github.io/how_convolutional_neural_networks_work.html

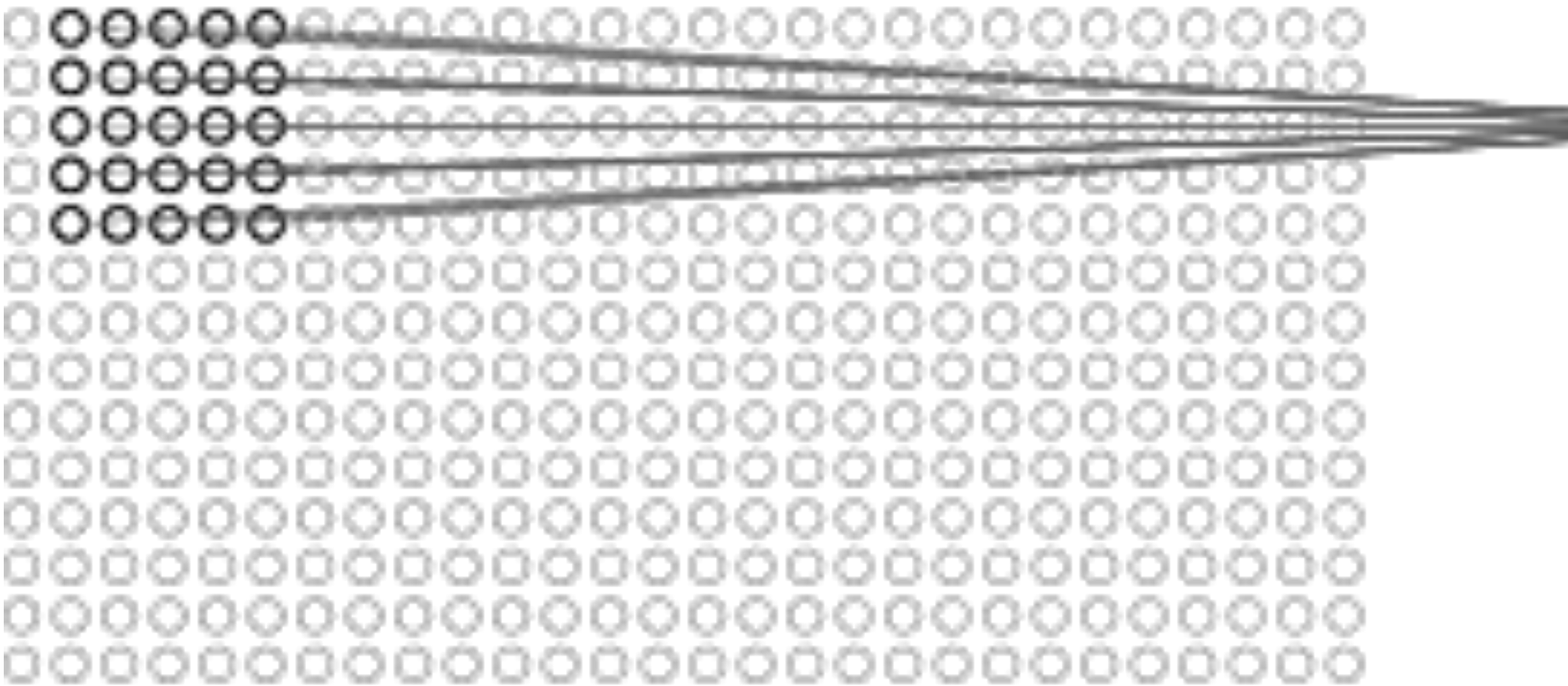
Shared weights and biases



Shared weights and biases

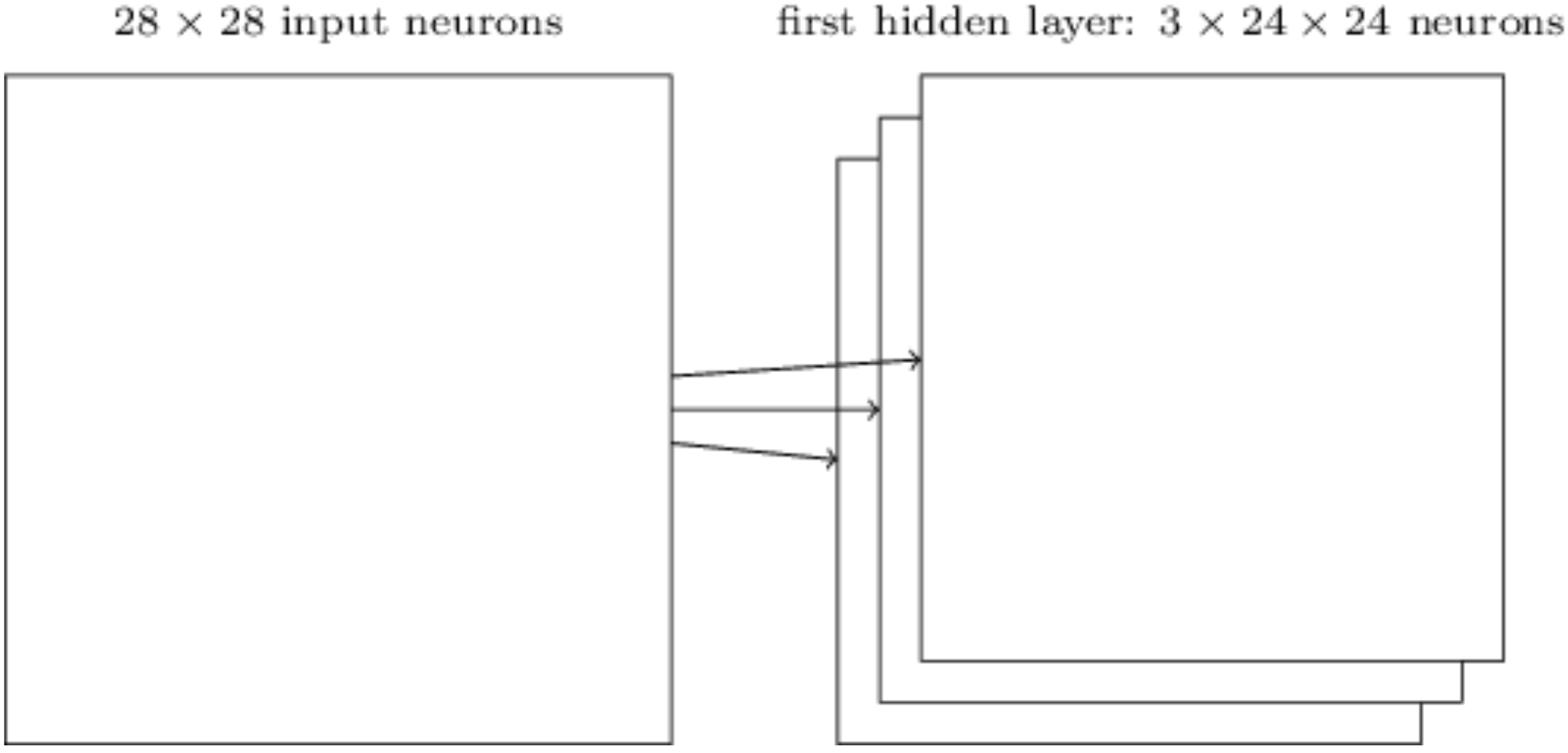


Strides

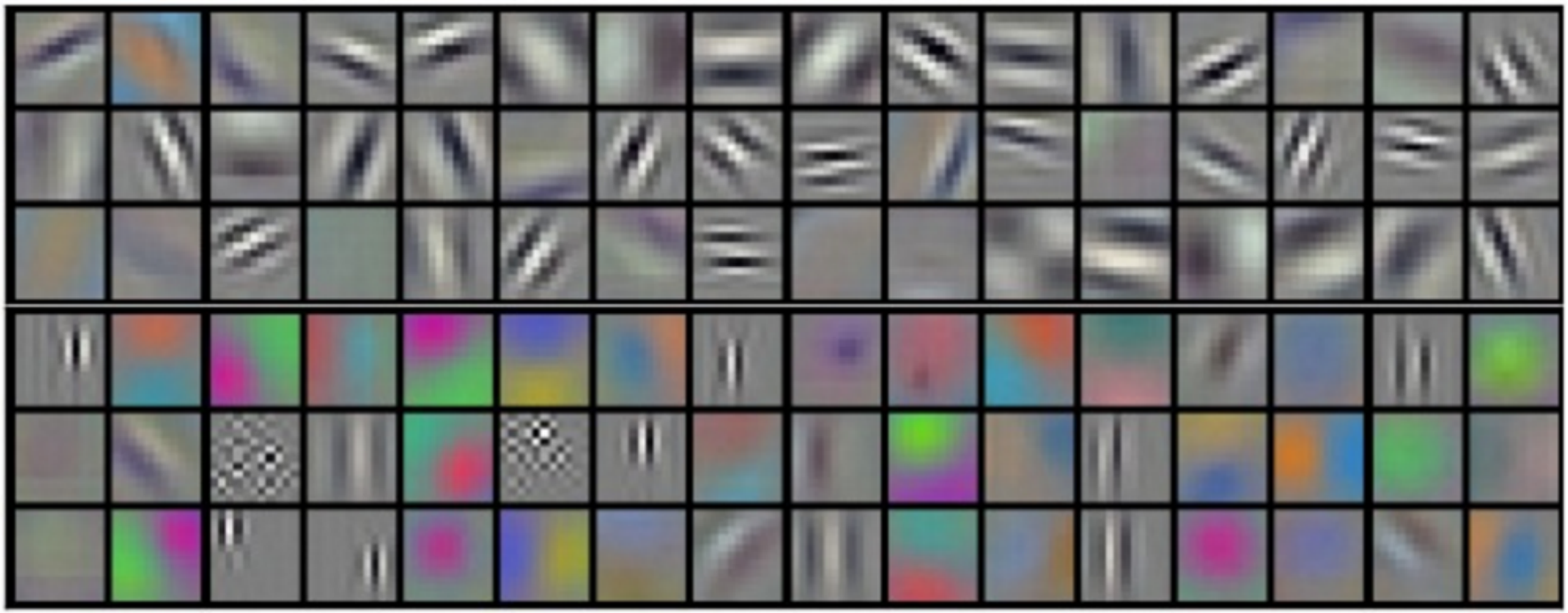


Strides

Filter maps

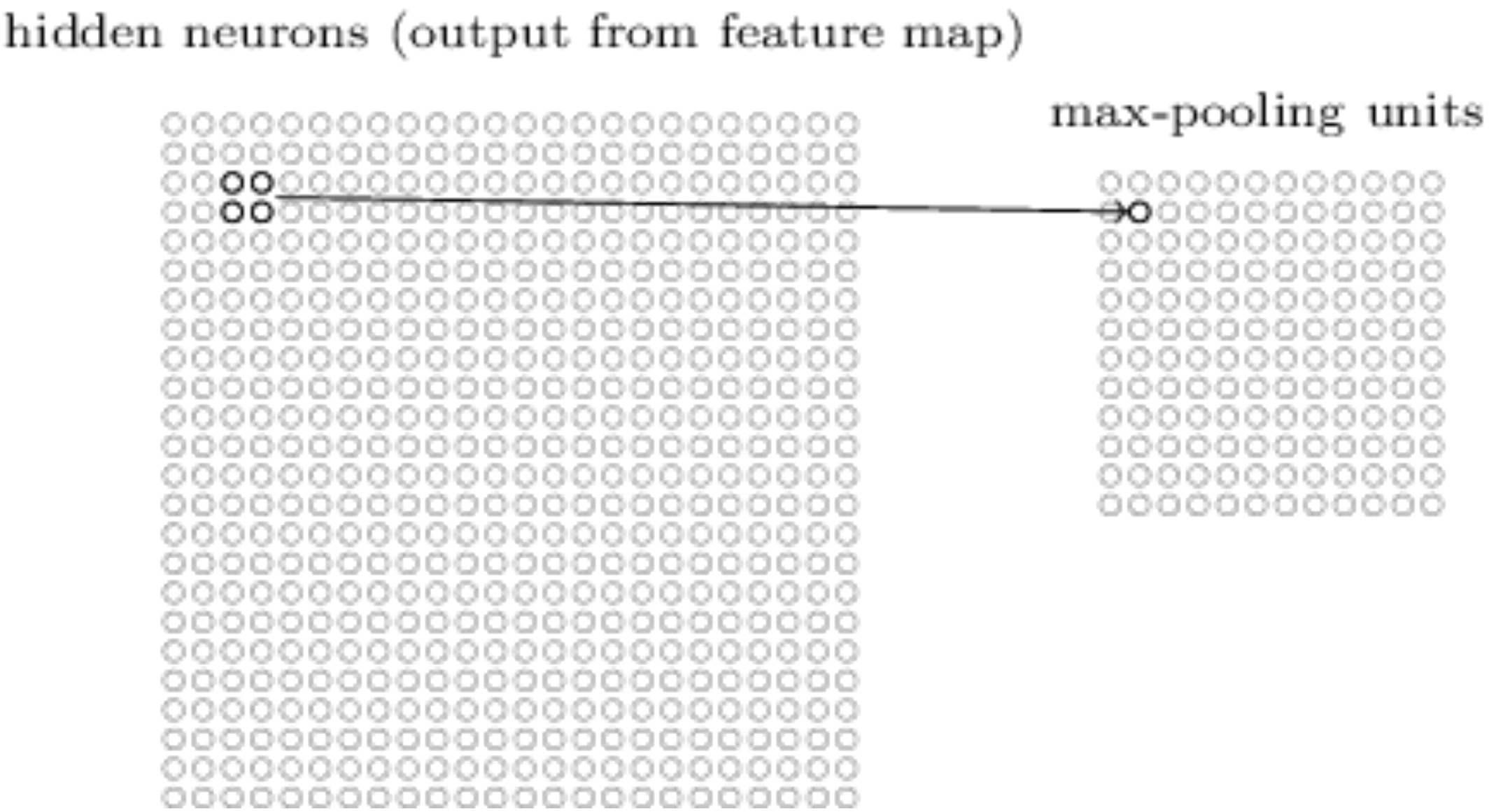


Learned Filters

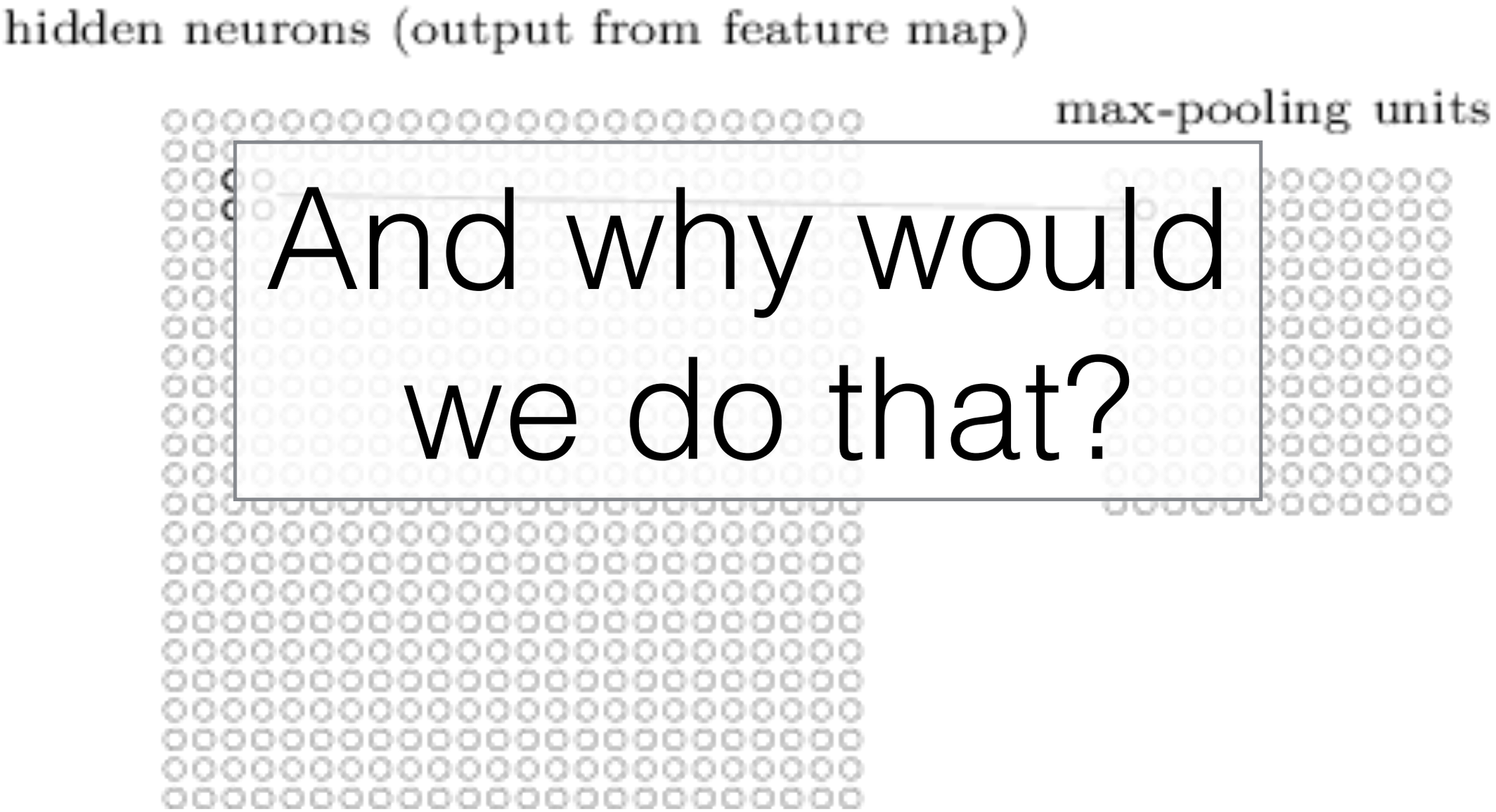


http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

Pooling



Pooling



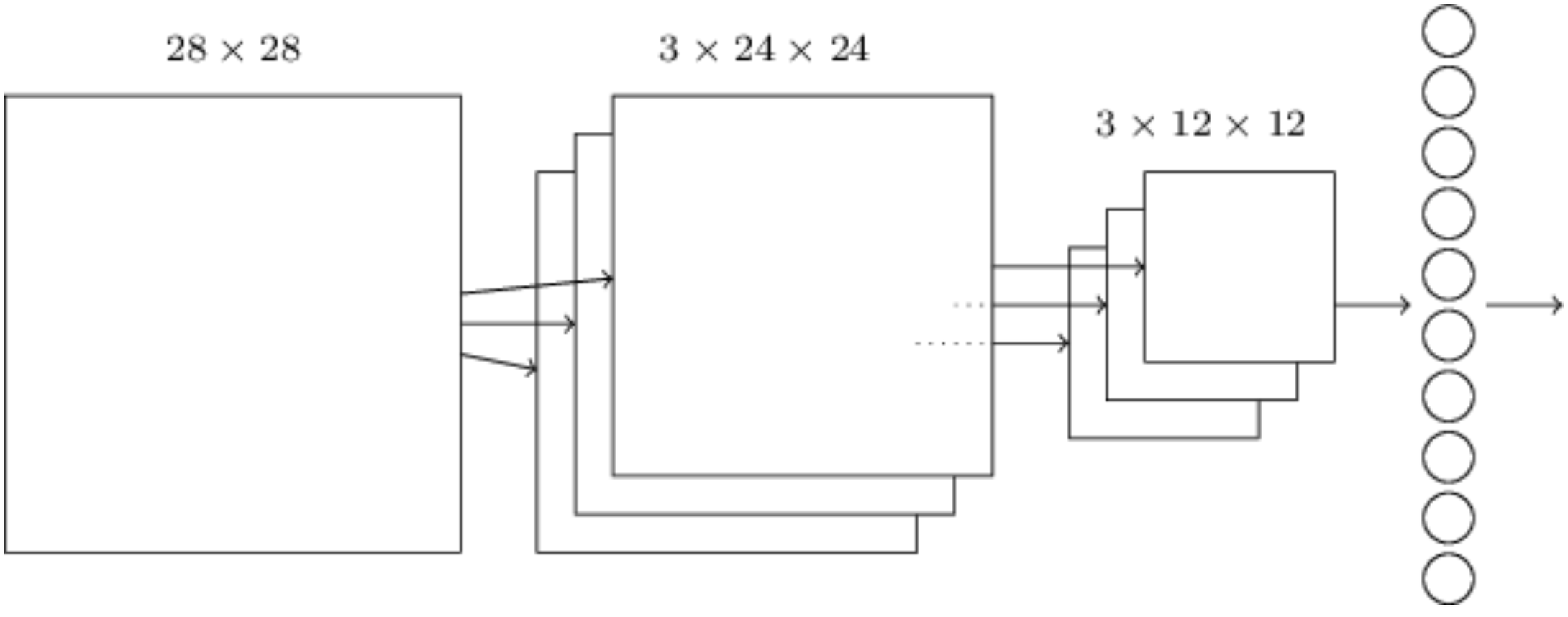
And why would we do that?

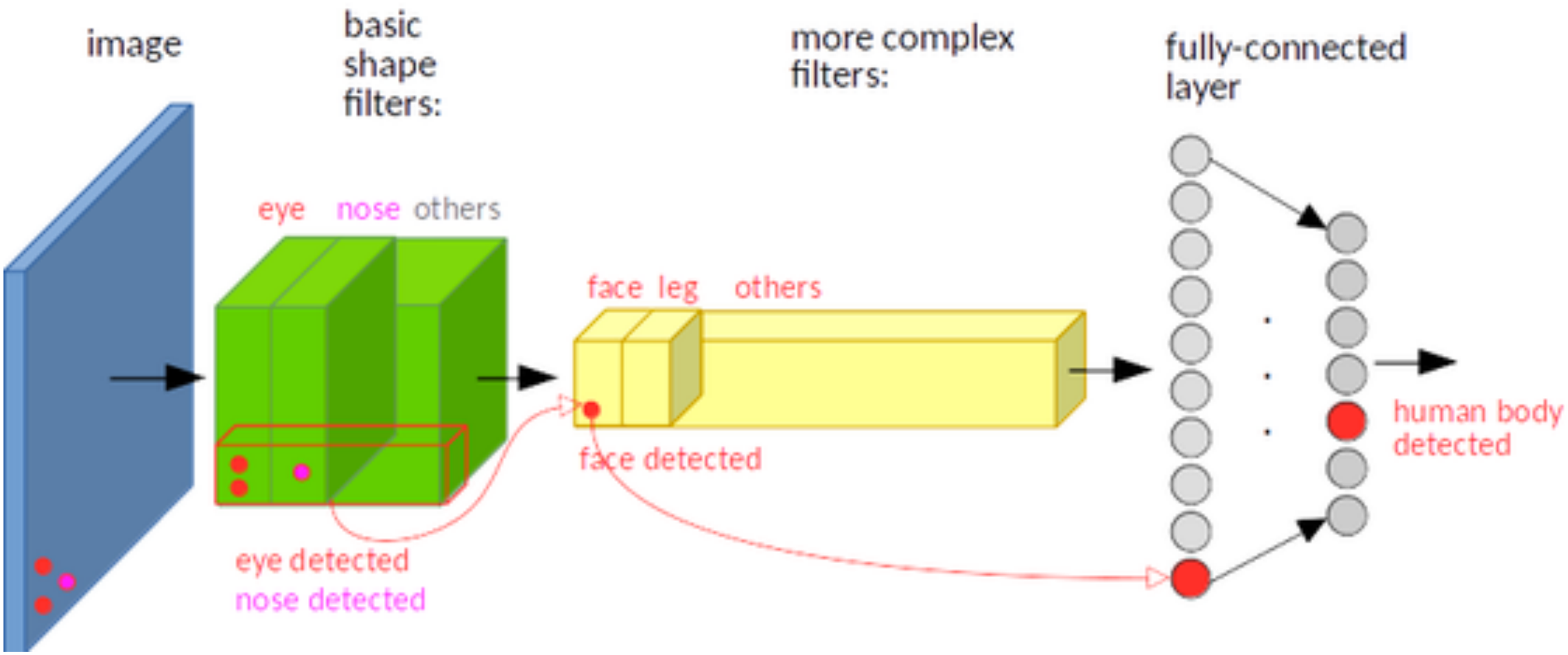
Pooling

And why would we do that?

=> local invariance & reduced parameters

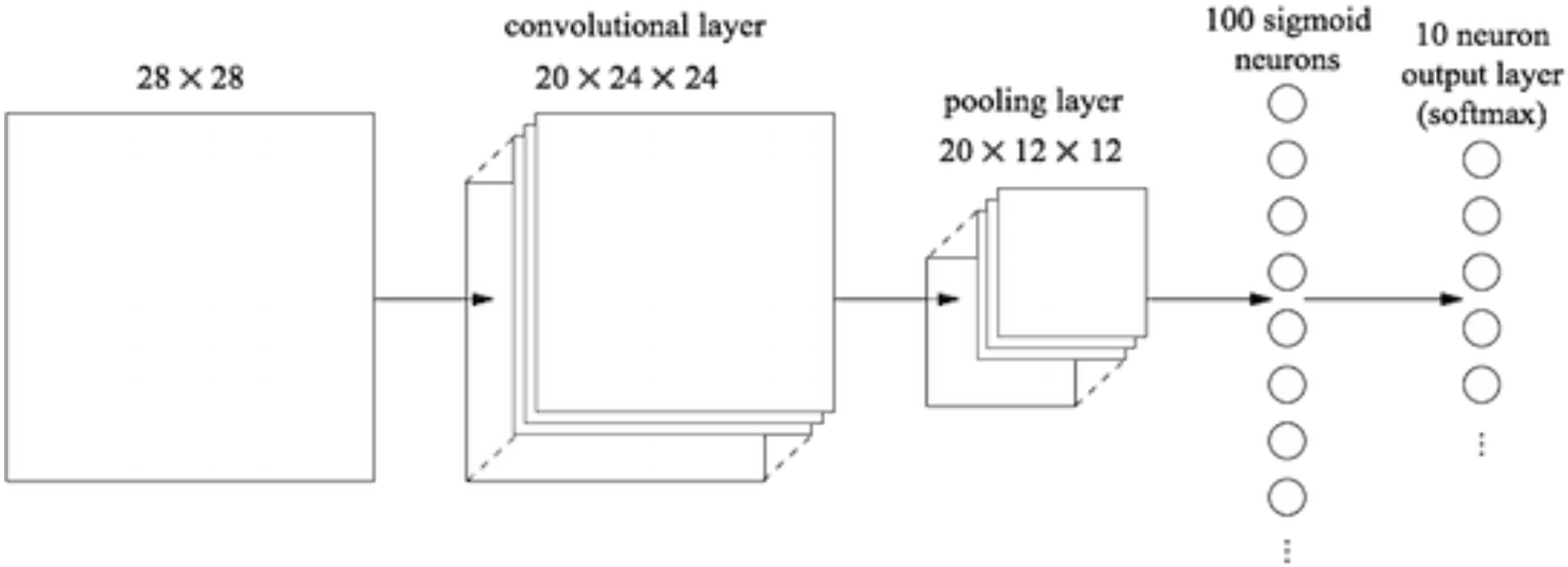
Convolutional Neural Networks



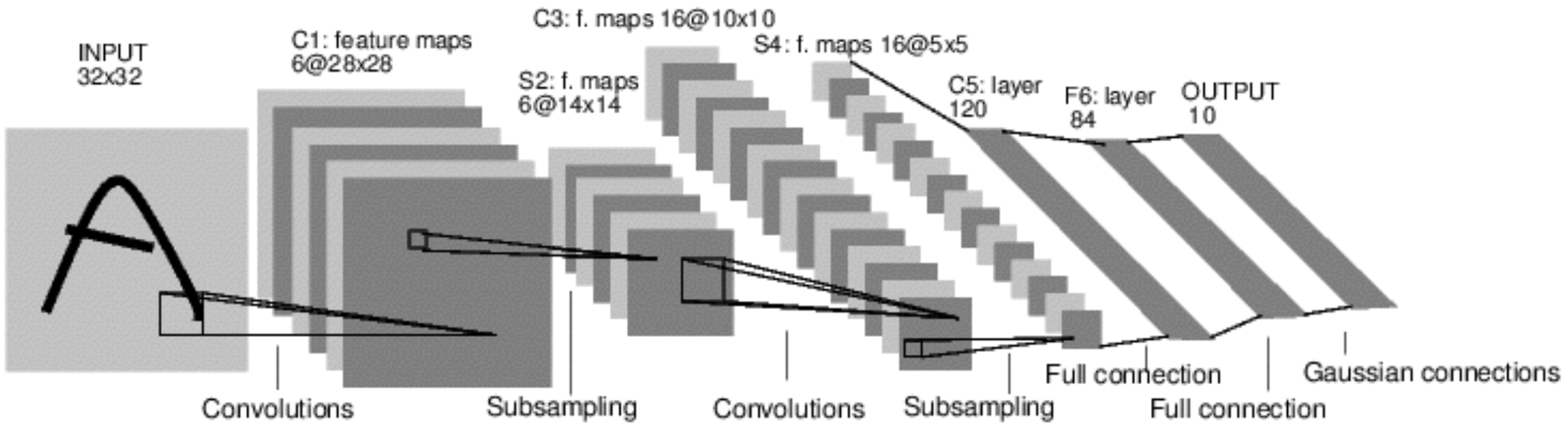


www.quora.com/How-is-a-convolutional-neural-network-able-to-learn-invariant-features/answer/Jean-Da-Rolt

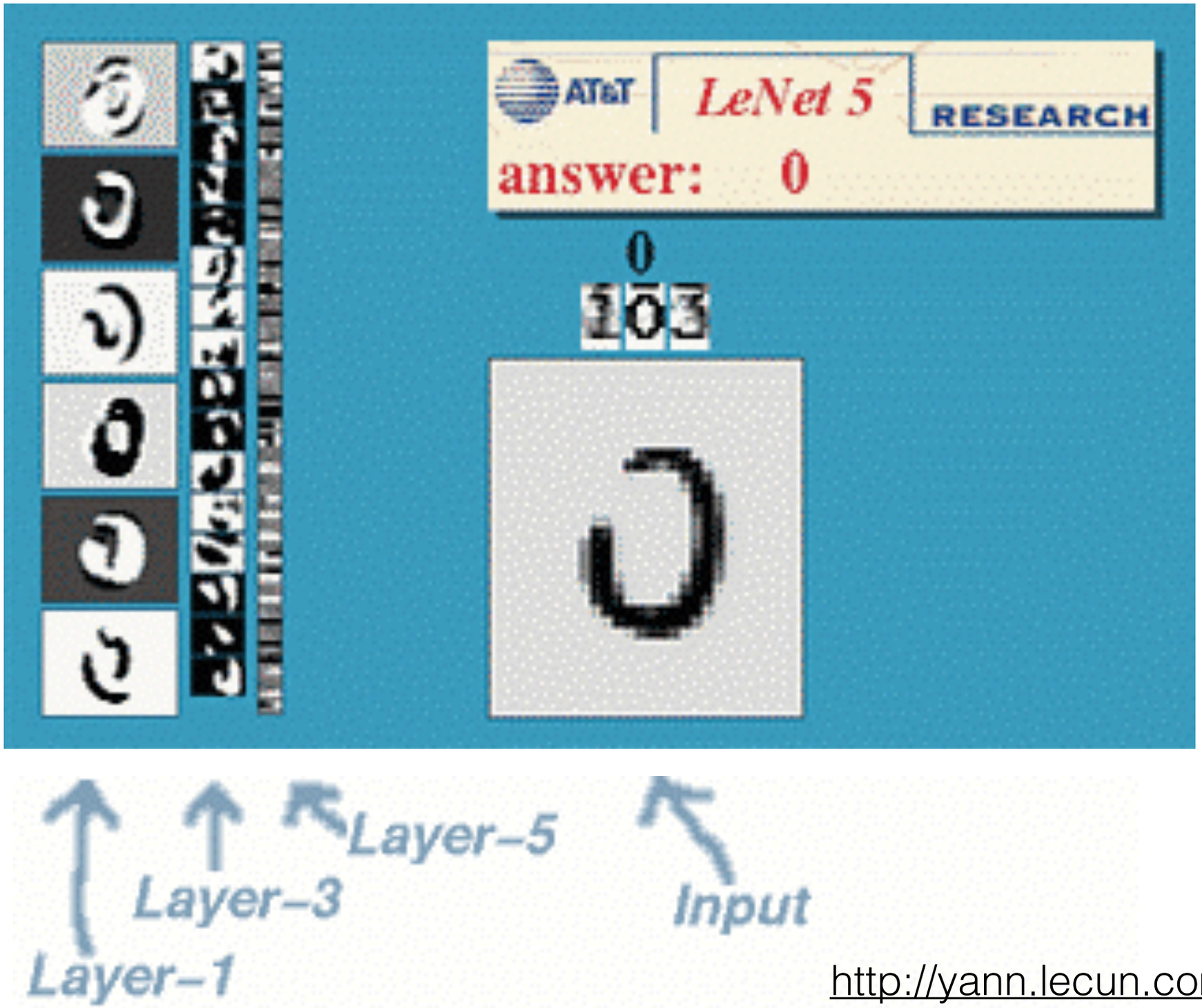
Convolutional Neural Networks



LeNet5 1998

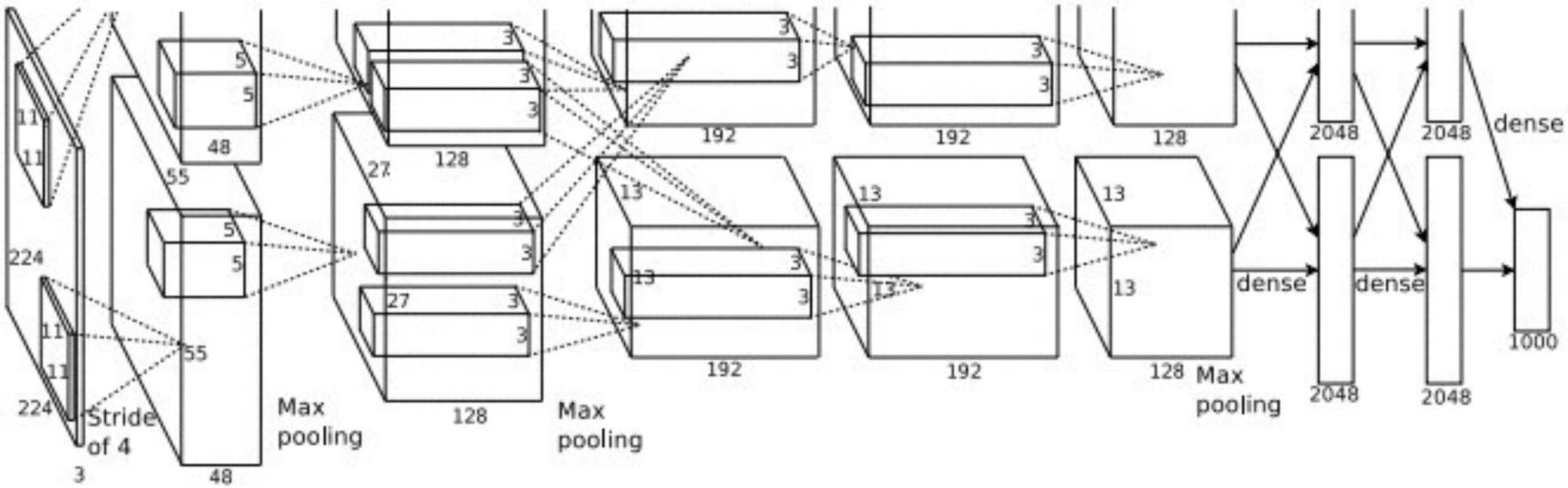

INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

LeNet5 1998



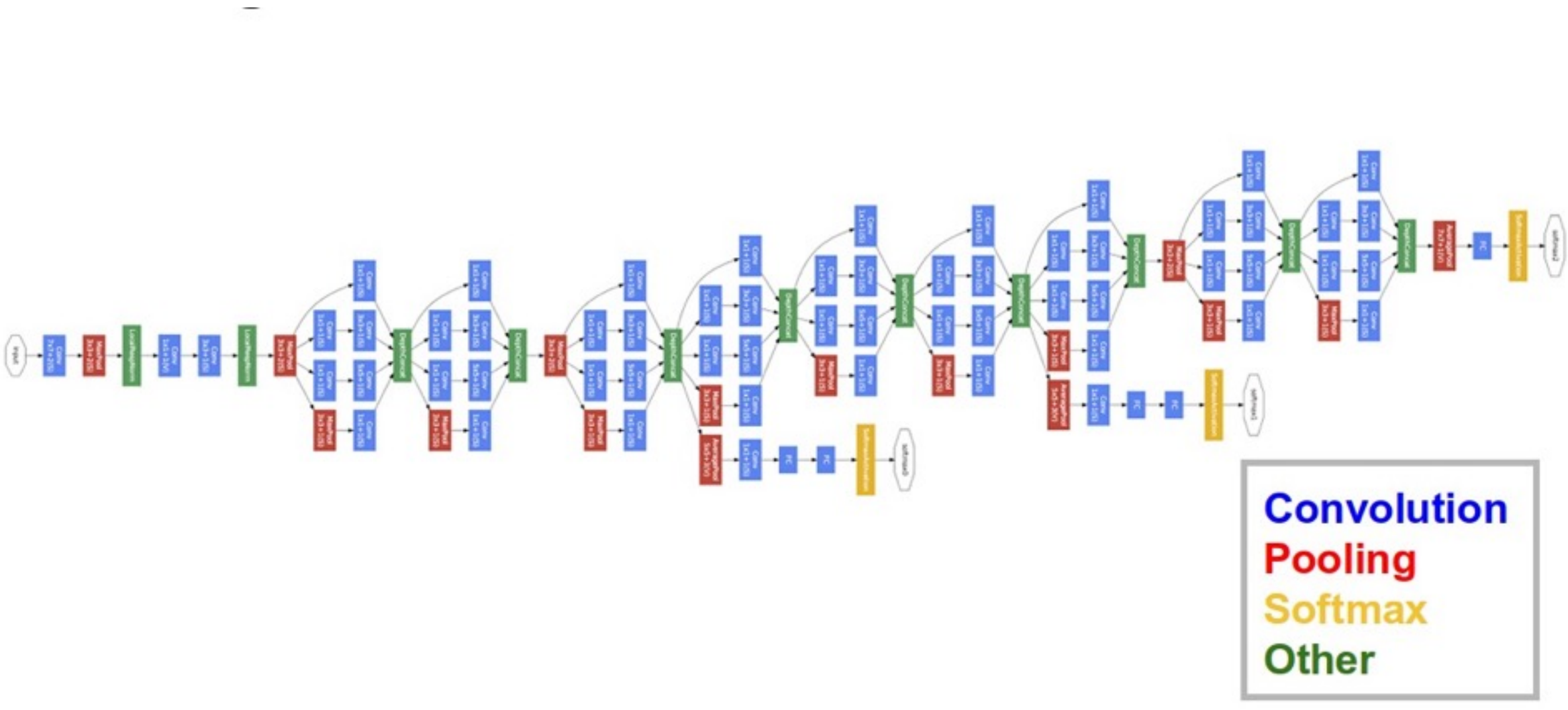
http://yann.lecun.com/exdb/lenet/

Alexnet 2012

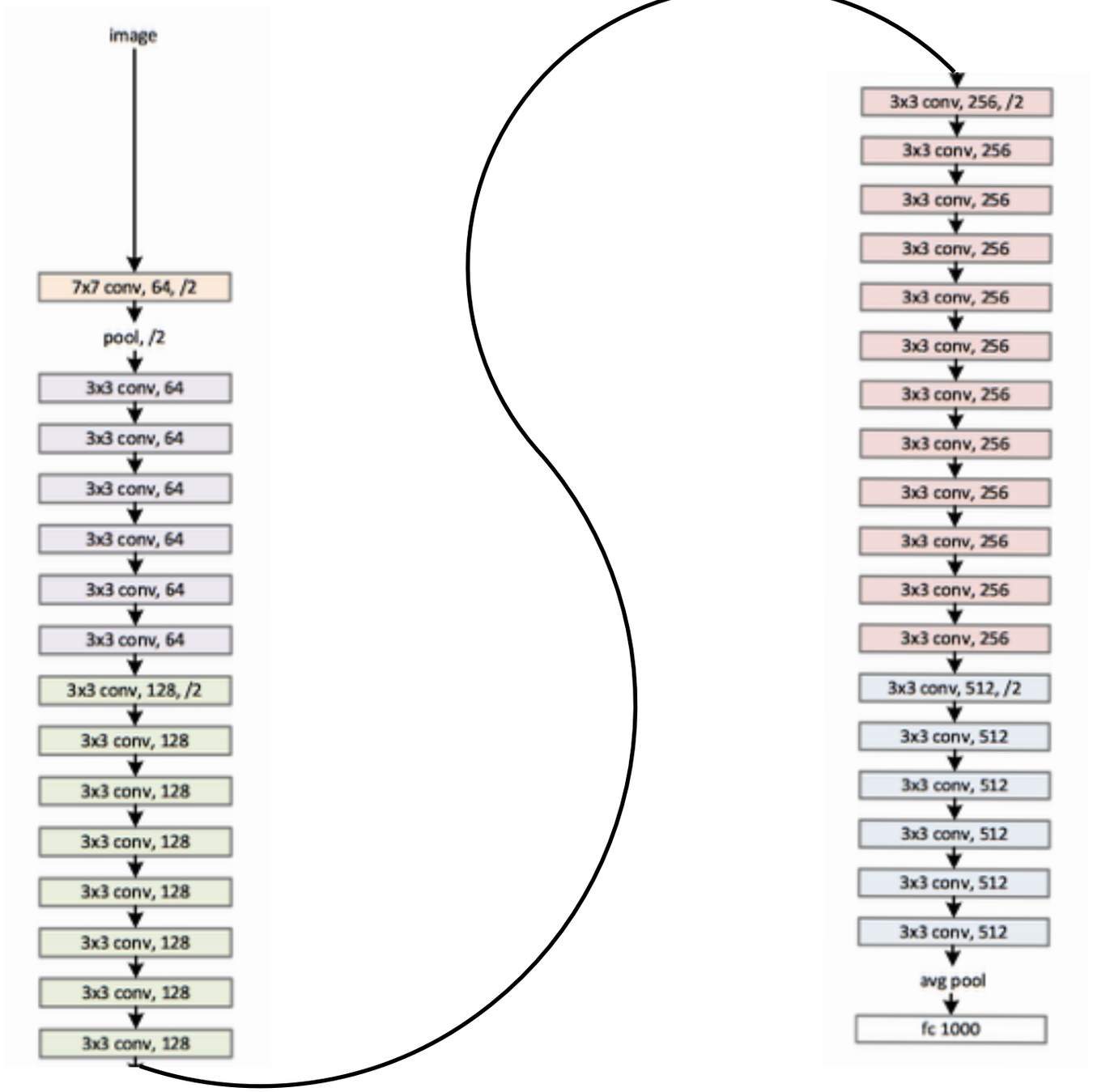


http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

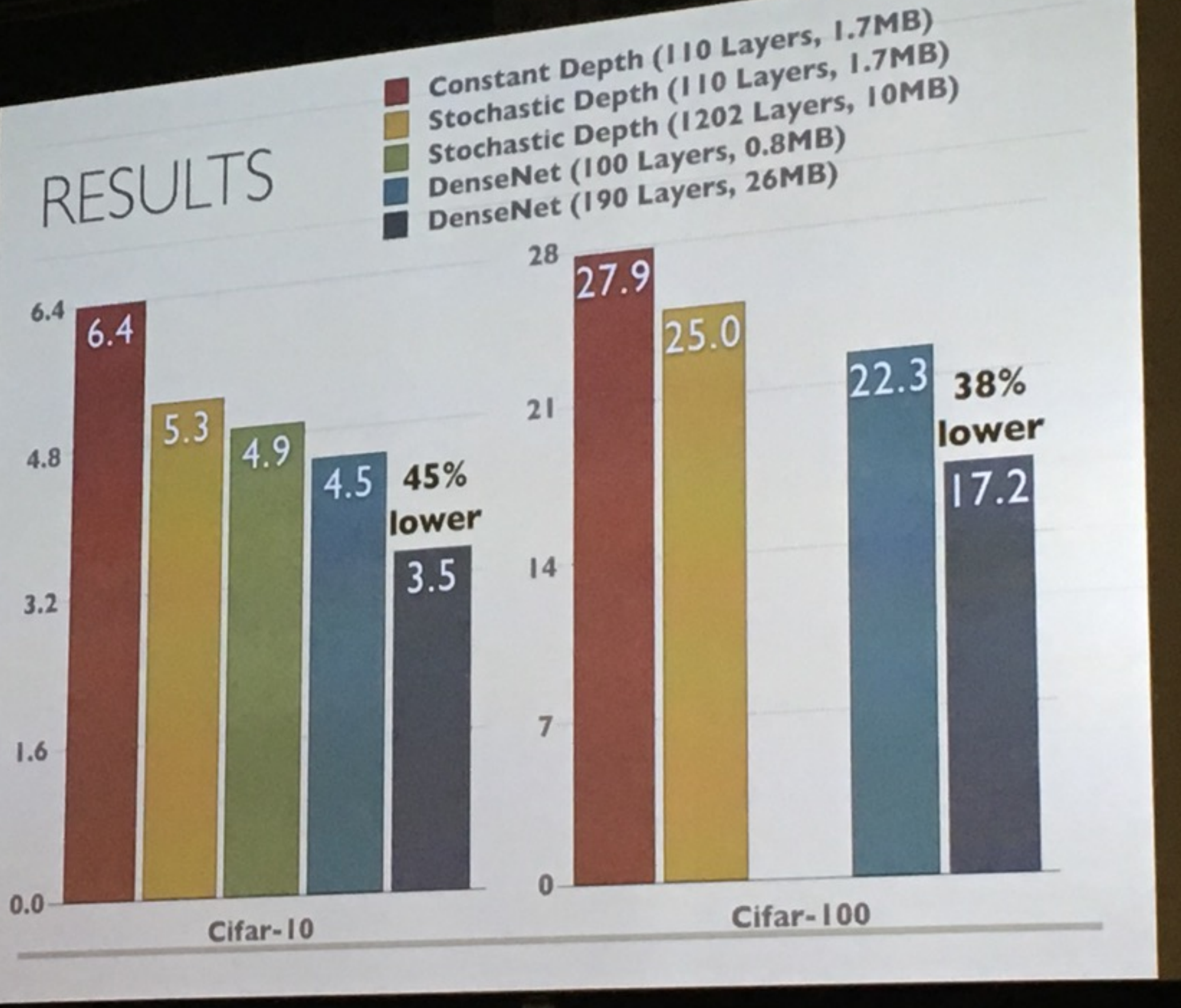
Google LeNet 2014



Microsofts Deep Residual Networks 2015



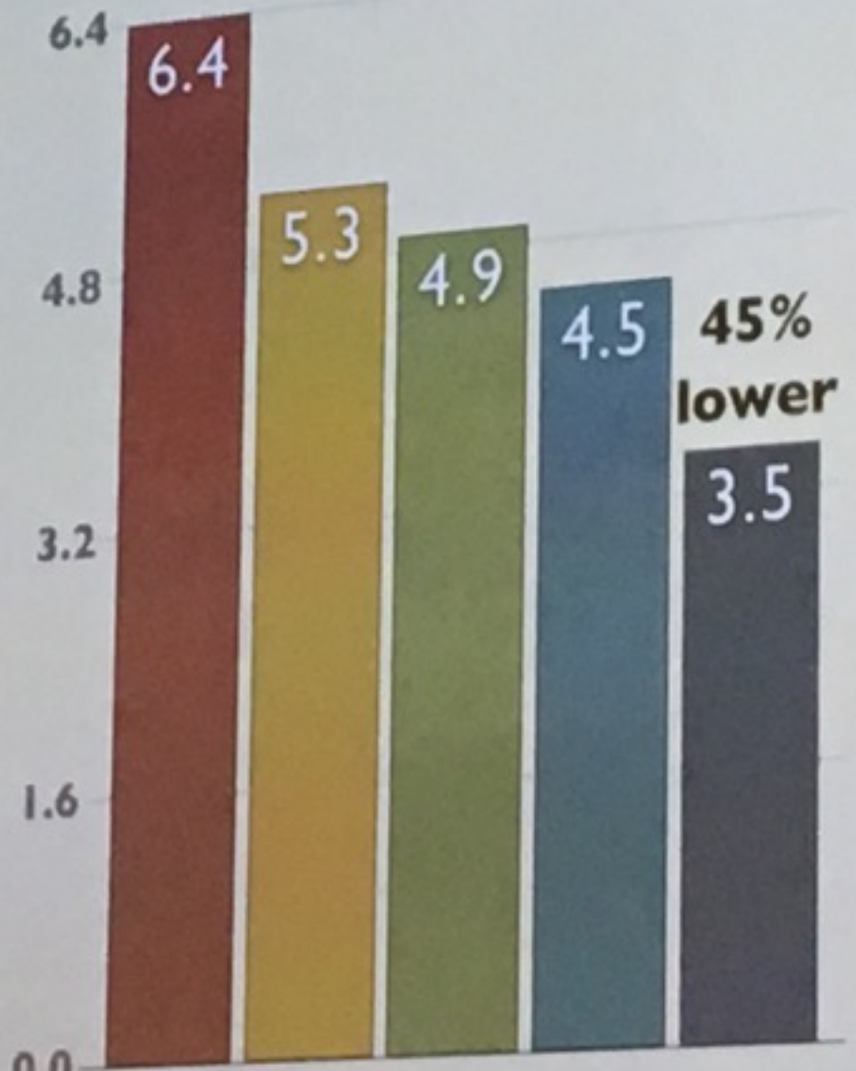
RESULTS

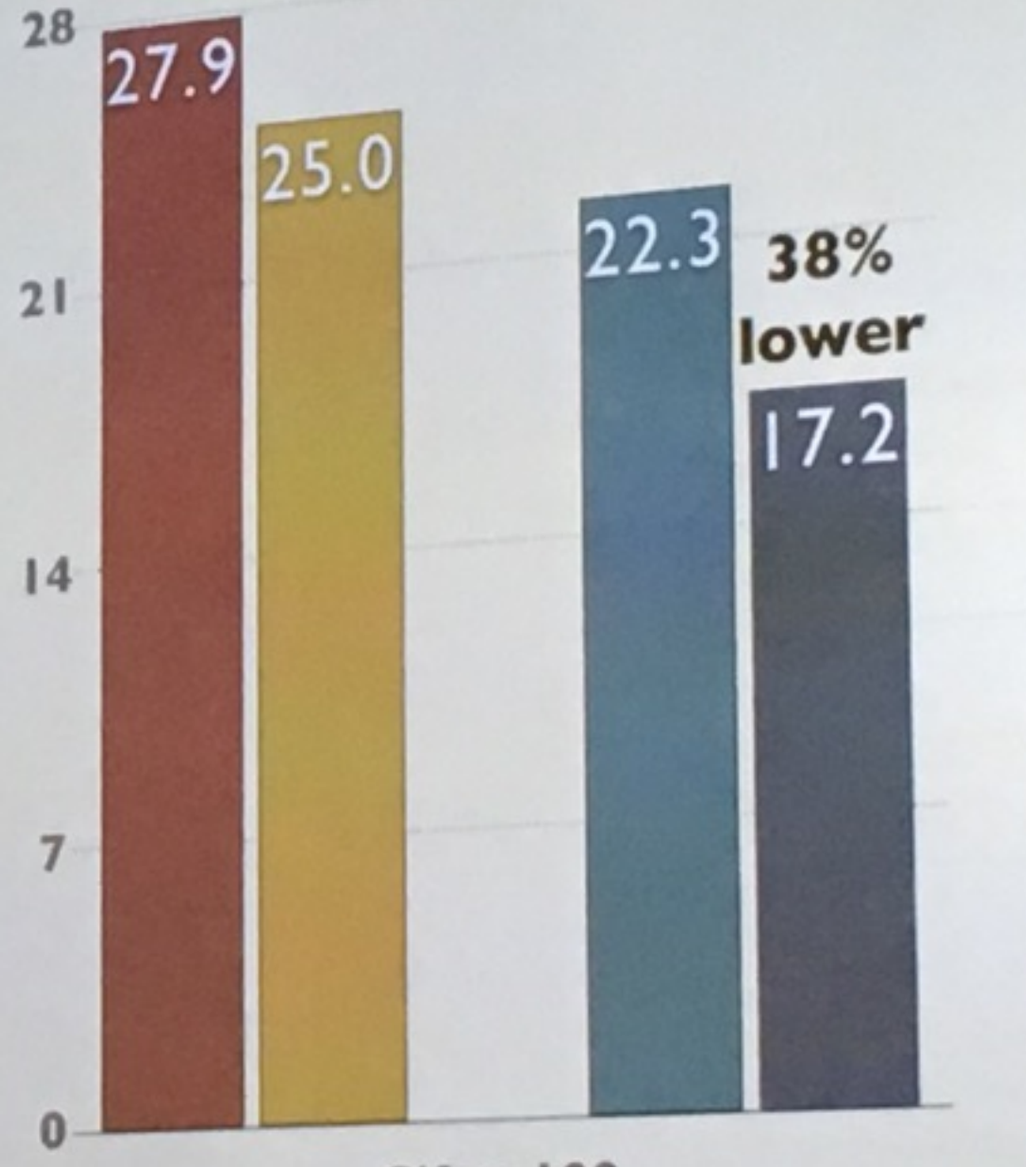


Densely Connected Convolutional Networks (NIPS 2016)

RESULTS

Constant Depth (110 Layers, 1.7MB)
Stochastic Depth (110 Layers, 1.7MB)
Stochastic Depth (1202 Layers, 10.2MB)
DenseNet (100 Layers, 0.8MB)
DenseNet (190 Layers, 26MB)





Densely Connected Convolutional Networks (NIPS 2016)

Frameworks

`Hello World` in CUDA

```
#include <stdio.h>
const int N = 16;
const int blocksize = 16;
__global__
void hello(char *a, int *b)
{
    a[threadIdx.x] += b[threadIdx.x];
}
int main()
{
    char a[N] = "Hello \0\0\0\0\0\0";
    int b[N] = {15, 10, 6, 0, -11, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
    char *ad;
    int *bd;
    const int csize = N*sizeof(char);
    const int isize = N*sizeof(int);
    printf("%s", a);
    cudaMalloc( (void**)&ad, csize );
    cudaMalloc( (void**)&bd, isize );
    cudaMemcpy( ad, a, csize,
cudaMemcpyHostToDevice );
cudaMemcpy( bd, b, isize,
cudaMemcpyHostToDevice );

    dim3 dimBlock( blocksize, 1 );
    dim3 dimGrid( 1, 1 );
    hello<<<dimGrid,
dimBlock>>>(ad, bd);
    cudaMemcpy( a, ad, csize,
cudaMemcpyDeviceToHost );
    cudaFree( ad );
    cudaFree( bd );

    printf("%s\n", a);
    return EXIT_SUCCESS;
}
```

http://computer-graphics.se/hello-world-for-cuda.html

How to decide what Deep Learning library to use?

- Features
  - `speciality` layers
    - most recent loss and activation functions
    - training ‘hacks’ (dropout)
    - helper functions
- Abstraction
- Speed
  - CPU/GPU speed
    - multi GPU support
- Ease of customization
- Base language
  - Python, C++, Java, Rust(!)…

Deep learning libraries: Accumulated GitHub metrics

Aggregate popularity (30•contrib + 10•issues + 5•forks)•1e-3		
#1:	172.29	tensorflow/tensorflow
#2:	89.78	BVLC/caffe
#3:	69.70	fchollet/keras
#4:	53.09	dmlc/mxnet
#5:	38.23	Theano/Theano
#6:	29.86	deeplearning4j/deeplearning4j
#7:	27.99	Microsoft/CNTK
#8:	17.36	torch/torch7
#9:	14.43	baidu/paddle
#10:	13.10	pfnet/chainer
#11:	12.37	NVIDIA/DIGITS
#12:	10.42	tflearn/tflearn
#13:	9.20	pytorch/pytorch

François Chollet @fchollet · Feb 11

Time for an update: what does the deep learning library landscape look like, seen from GitHub?
pic.twitter.com/QDZyvLrYBd

12 290 386

Deep learning libraries: Accumulated GitHub metrics

Aggregate popularity (30•contrib + 10•issues + 5•forks)•1e-3		
#1:	172.29	tensorflow/tensorflow
#2:	89.78	BVLC/caffe
#3:	69.70	fchollet/keras
#4:	53.09	dmlc/mxnet
#5:	38.23	Theano/Theano
#6:	29.86	deeplearning4j/deeplearning4j
#7:	27.99	Microsoft/CNTK
#8:	17.36	torch/torch7
#9:	14.43	baidu/paddle
#10:	13.10	pfnet/chainer
#11:	12.37	NVIDIA/DIGITS
#12:	10.42	tflearn/tflearn
#13:	9.20	pytorch/pytorch

François Chollet @fchollet · Feb 11

Time for an update: what does the deep learning library landscape look like, seen from GitHub?
pic.twitter.com/QDZyvLrYBd

12 290 386

Tensor flow

- Features
  - `speciality` layers
    - most recent loss and activation functions ✅
    - training ‘hacks’ (dropout) ✅
    - helper functions ✅
- Abstraction
- Speed
  - CPU/GPU speed
    - multi GPU support ✅
- Ease of customization ✅
- Base language
  - Python ✅, C++, Java, Rust(!)…

Deep learning libraries: Accumulated GitHub metrics

Aggregate popularity (30•contrib + 10•issues + 5•forks)•1e-3		
#1:	172.29	tensorflow/tensorflow
#2:	89.78	BVLC/caffe
#3:	69.70	fchollet/keras
#4:	53.09	dmlc/mxnet
#5:	38.23	Thea…
#6:	29.86	…eplearning4j
#7:	27.99	…t/CNTK
#8:	1…	torch/torch7
#9:	1…	baidu/paddle
#10:	1…	pfnet/chainer
#11:	12.37	NVIDIA/DIGITS
#12:	10.42	tflearn/tflearn
#13:	9.20	pytorch/pytorch

Why it is called Tensorflow?

François Chollet @fchollet · Feb 11

Time for an update: what does the deep learning library landscape look like, seen from GitHub?
pic.twitter.com/QDZyvLrYBd

12 290 386

Init

```
import tensorflow as tf
sess = tf.InteractiveSession()
```

Setup variables

```
x = tf.placeholder(tf.float32,
    shape=[None, 784])
y_ = tf.placeholder(tf.float32,
    shape=[None, 10])
W = tf.Variable(tf.zeros([784,10]))
b = tf.Variable(tf.zeros([10]))

sess.run(tf.global_variables_initializer())
```

Define the network

```
y = tf.matmul(x,W) + b
softmax = tf.nn.\
    softmax_cross_entropy_with_logits(y_, y)
cross_entropy = tf.reduce_mean(softmax)
GD = tf.train.GradientDescentOptimizer
train_step = GD().minimize(cross_entropy)
```

Train

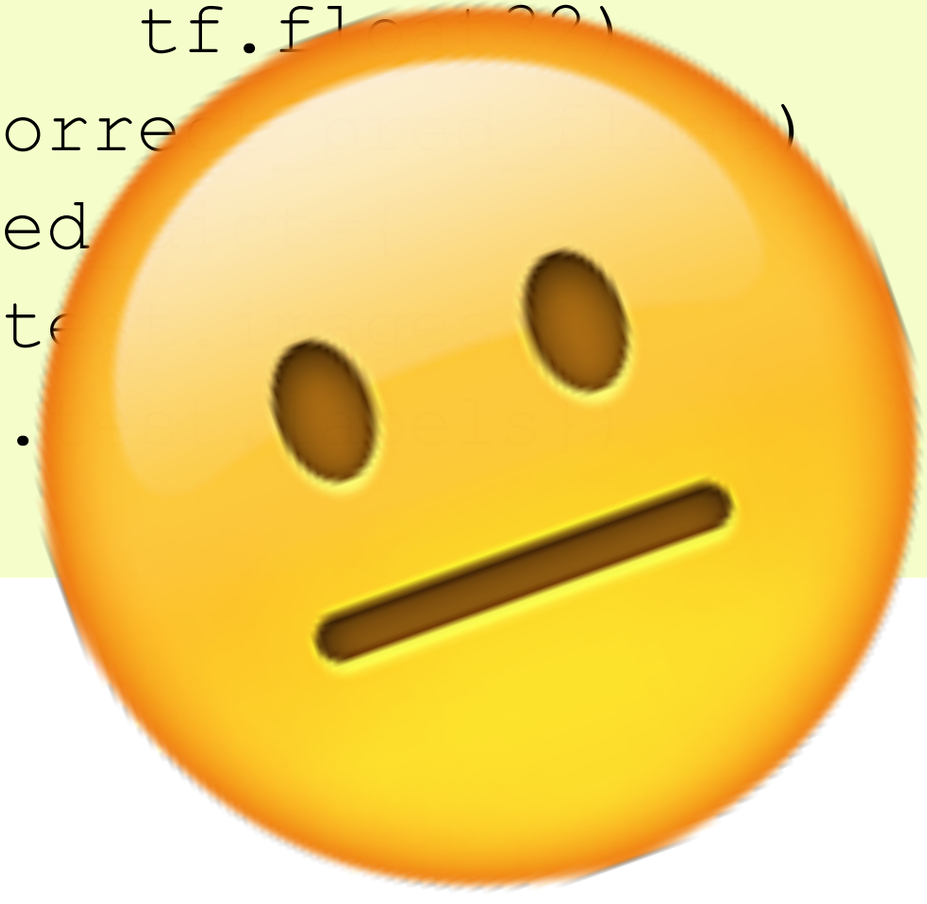
```
GD = tf.train.GradientDescentOptimizer
train_step = GD().minimize(cross_entropy)
for _ in range(1000):
  batch = mnist.train.next_batch(100)
  train_step.run(feed_dict={x: batch[0],
                            y_: batch[1]})
```

Evaluate

```
correct_pred = tf.equal(tf.argmax(y,1),
                        tf.argmax(y_,1))
correct_pred_float = tf.cast(correct_pred,
                             tf.float32)
acc_fun = tf.reduce_mean(correct_pred_float)
accuracy = acc_fun.eval(feed_dict={
                        x: mnist.test.images,
                        y_: mnist.test.labels})
print accuracy
```

Evaluate

```
correct_pred = tf.equal(tf.argmax(y,1),
                        tf.argmax(y_,1))
correct_pred_float = tf.cast(correct_pred,
                             tf.float32)
acc_fun = tf.reduce_mean(correct_pred_float)
accuracy = acc_fun.eval(feed_dict={
                        x: mnist.test.images,
                        y_: mnist.test.labels})
print accuracy
```



Deep learning libraries: Accumulated GitHub metrics

Aggregate popularity (30•contrib + 10•issues + 5•forks)•1e-3		
#1:	172.29	tensorflow/tensorflow
#2:	89.78	BVLC/caffe
#3:	69.70	fchollet/keras
#4:	53.09	dmlc/mxnet
#5:	38.23	Theano/Theano
#6:	29.86	deeplearning4j/deeplearning4j
#7:	27.99	Microsoft/CNTK
#8:	17.36	torch/torch7
#9:	14.43	baidu/paddle
#10:	13.10	pfnet/chainer
#11:	12.37	NVIDIA/DIGITS
#12:	10.42	tflearn/tflearn
#13:	9.20	pytorch/pytorch

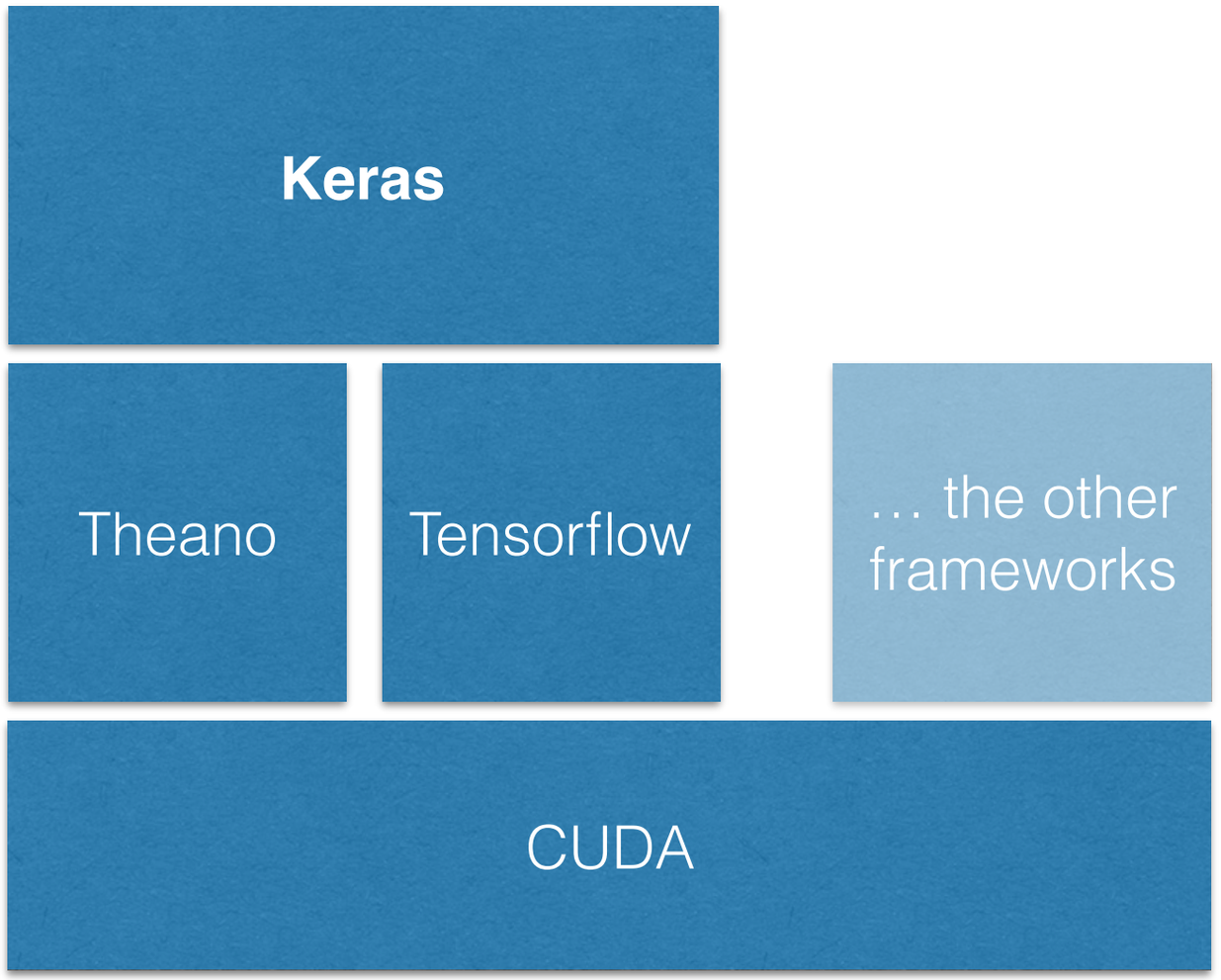
François Chollet @fchollet · Feb 11

Time for an update: what does the deep learning library landscape look like, seen from GitHub?
pic.twitter.com/QDZyvLrYBd

12 290 386

Keras

- NN framework for fast prototyping
- High level
- It supports Tensorflow AND Theano
- Soon part of the official TensorFlow



Tensor flow

- Features
  - `speciality` layers
    - most recent loss and activation functions ✅
    - training ‘hacks’ (dropout) ✅
    - helper functions ✅
- Abstraction ✅

- Speed
  - CPU/GPU speed
    - multi GPU support ✅
- Ease of customization ✅
- Base language
  - Python ✅, C++, Java, Rust(!)…

=> We will use Keras

Init

TF

```
import tensorflow as tf
sess = tf.InteractiveSession()
```

K

```
from keras.models import Sequential
from keras.layers.core import Dense, Dropout,
     Activation
from keras.optimizers import SGD
```

Setup variables

TF

```
x = tf.placeholder(tf.float32,
    shape=[None, 784])
y_ = tf.placeholder(tf.float32,
    shape=[None, 10])
W = tf.Variable(tf.zeros([784,10]))
b = tf.Variable(tf.zeros([10]))

sess.run(tf.global_variables_initializer())
```

K

```
# not necessary
```

Define the network

TF

```
y = tf.matmul(x,W) + b
softmax = tf.nn.\
    softmax_cross_entropy_with_logits(y_, y)
cross_entropy = tf.reduce_mean(softmax)
GD = tf.train.GradientDescentOptimizer
train_step = GD().minimize(cross_entropy)
```

K

```
model = Sequential()
model.add(Dense(10, input_shape=(784,)))
model.add(Activation('sigmoid'))
model.compile(loss='categorical_crossentropy',
    optimizer=SGD(), metrics=['accuracy'])
```

Train

TF

```
for _ in range(1000):
  batch = mnist.train.next_batch(100)
  train_step.run(feed_dict={x: batch[0],
                            y_: batch[1]})
```

K

```
history = model.fit(X_train, Y_train,
  batch_size=batch_size,
  nb_epoch=nb_epoch, verbose=1,
  validation_data=(X_test, Y_test))
```

Evaluate

TF

```
correct_pred = tf.equal(tf.argmax(y,1),
                        tf.argmax(y_,1))
correct_pred_float = tf.cast(correct_pred,
                             tf.float32)
acc_fun = tf.reduce_mean(correct_pred_float)
accuracy = acc_fun.eval(feed_dict={
                        x: mnist.test.images,
                        y_: mnist.test.labels})
print accuracy
```

K

```
score = model.evaluate(X_test, Y_test, verbose=0)
print 'Test score:', score[0]
print 'Test accuracy:', score[1]
```

Help, my neural network is not learning!

What ‘Advanced’ techniques do we already talked about?

- Softmax
- Better weight initialization
- L2 Weight Regularization

Broad strategy

Simplify, simplify, simplify!

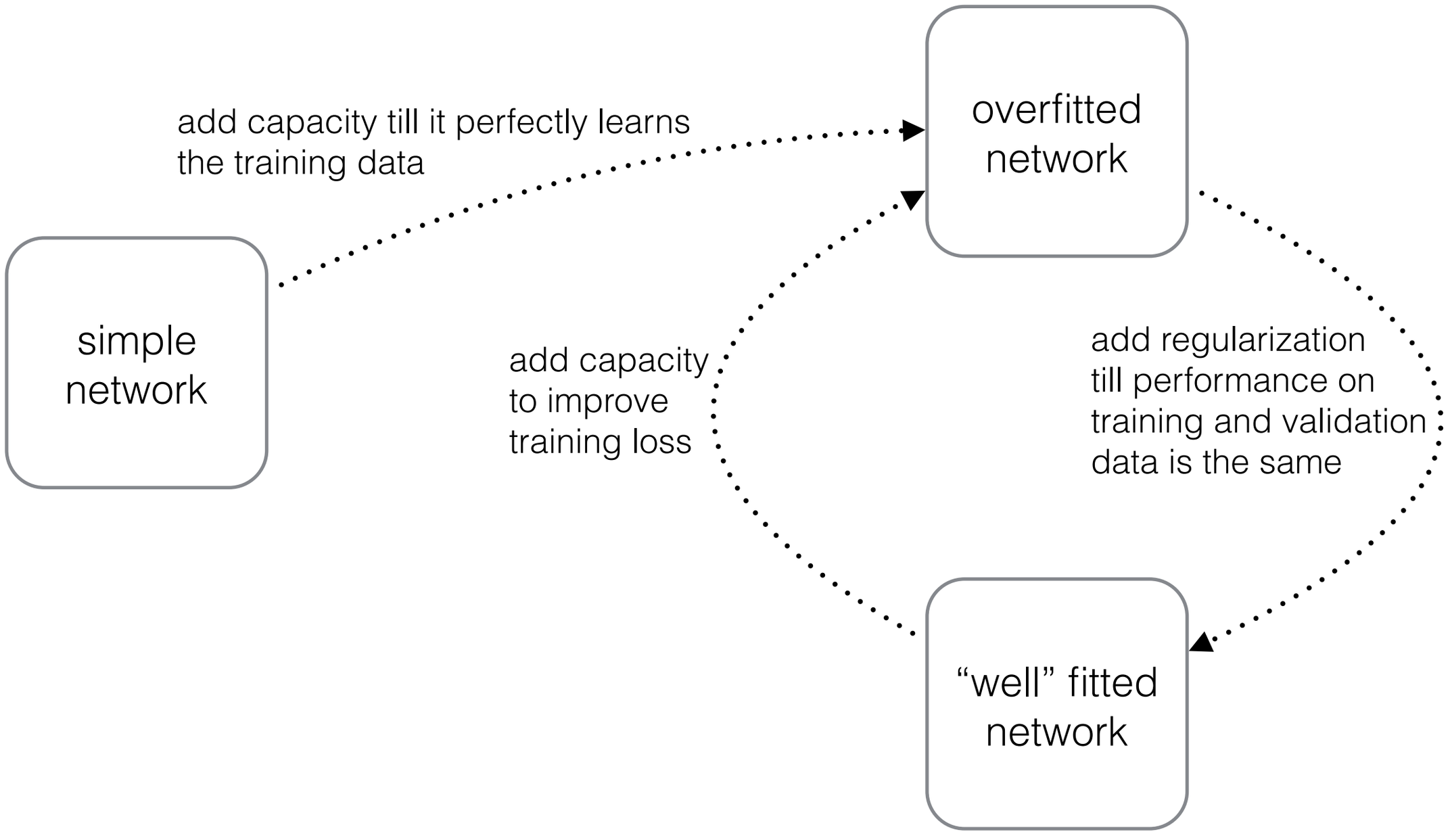
- Use smaller network
- Use subset of classes
- Monitor progress more often with less data

➡ Fail fast and try new parameter

What doesn't make a network simpler?

- outdated activation functions
- 'wrong' loss
- vanilla weight initialization

Regularization and network capacity

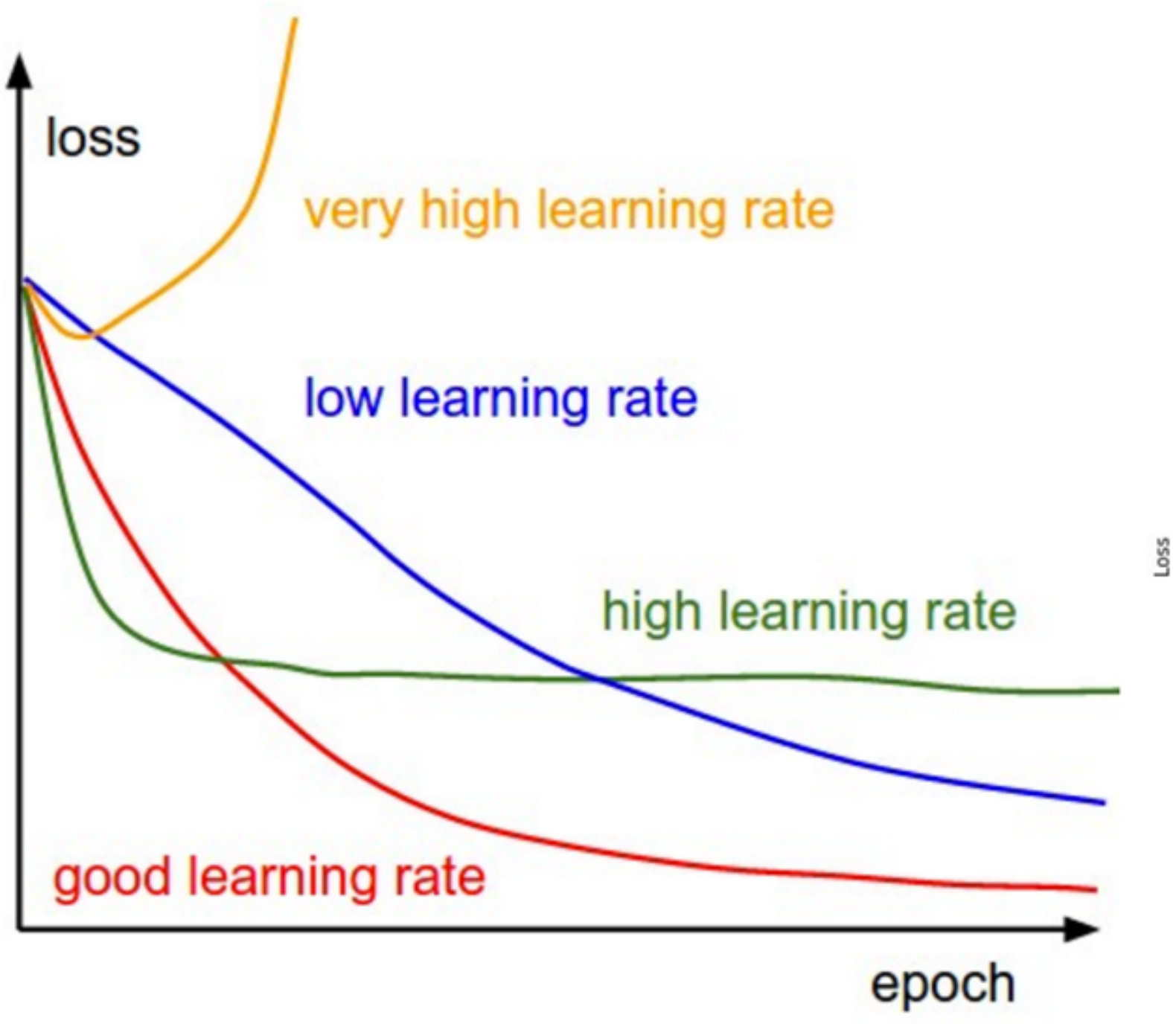


Learning rate

1. Try different magnitudes [1, 0.1, 10, 0.01, 100, …]
2. Take the biggest learning rate that doesn’t ‘explode’ the network.
3. Half it

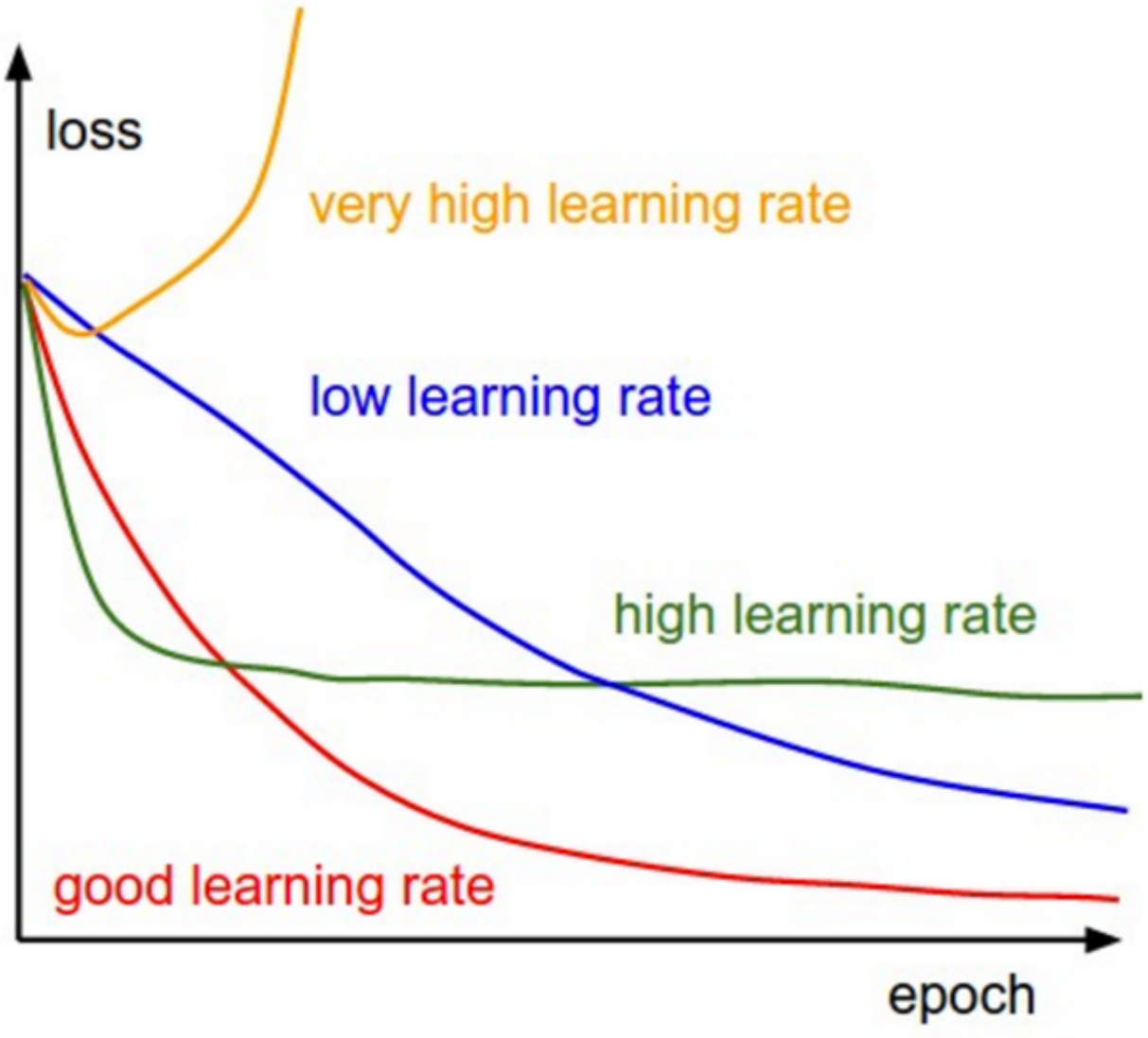
Detour: reading the loss ‘function’

Reading the loss ‘function’

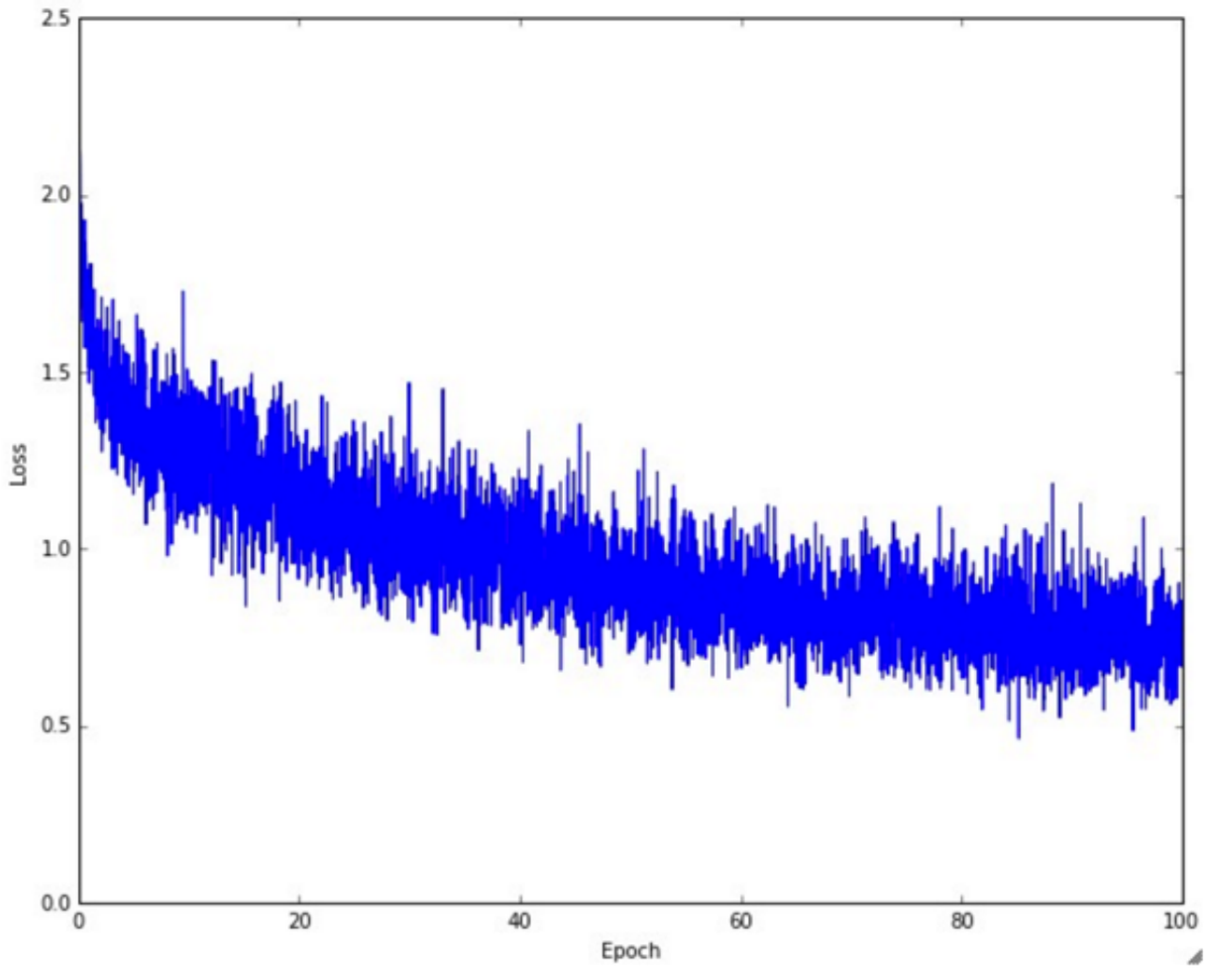


http://cs231n.github.io/neural-networks-3/

Reading the loss ‘function’



realistic loss function



http://cs231n.github.io/neural-networks-3/

Mini-batch size

Batch size 1

- also called online learning
- weights are updated often

Batch size 1< M< `n_complete`

- loss is computed on ‘outdated’ weights
- allows to profit of the special optimizations of the hard- and software implementation

Rule of thumb: Use the largest batch size, that you can fit into RAM - usually >10 and <500
To validate: Plot validation accuracy over time (not epochs) and choose sweet point

Dependence:
Learning rate and batch size

Batch size and learning rate are anti `proportional` in a non-linear way.

Rule of thumb:

batch_size * X => learning_rate / X (or sqrt(X))

Layers

Fully connected layers: Keep getting smaller from input to output

Convolutional layers: Set filter size 3x3, tune number of filters. Go from few filters to many filters.

(Max) Pooling layers: 2x2 or 3x3 and increase number filters

Data

- needs to be normalized
  - normally [0..1]
- augmentations can be used to increase dataset
  - mirroring, cropping, skewing, rotating…
  - also functions as regularization

Data

- needs to be normalized
  - normally [0..1]

Why do we need to normalize??

- augmentations can be used to increase dataset
  - mirroring, cropping, skewing, rotating…
  - also functions as regularization

Train MNIST with Keras

Play around with:

- learning rate
- batch size
- network architecture

Try out:

- Softmax
- Better weight initialization
- L2 Weight Regularization
- Augmentation
- **See https://keras.io/**

`~/keras_mnist.ipynb`

Train MNIST with Keras

Play around with:

- learning rate
- batch size
- network architecture

Try out:

- Softmax
- Better weight initialization
- L2 Weight Regularization
- Augmentation
- **See https://keras.io/**

Set verbose=2 in model.fit()

```
~/keras_mnist.ipynb
```